Prompt Engineering in Large Language Models to solve quantitative Problems

PH312: Mini Project

Submitted by:

Sunidhi Prakash

210121054

Engineering Physics

Name of the Supervisor:

Dr. Girish Sampath Setlur

Department of Physics

Indian Institute of Technology Guwahati

Abstract:

The ability of large language models (LLMs) to solve complex problems has garnered significant attention in recent years. In particular, the application of LLMs in solving JEE Advanced questions presents a unique challenge due to the intricate nature of the problems. Current approaches, including the utilization of the JEEBench dataset, have shown limitations in accurately solving these questions, highlighting the need for more effective techniques. This project addresses the aforementioned challenge by focusing on advanced prompting techniques aimed at pushing LLMs to their full problem-solving potential. Leveraging concepts such as chain of thought and few-shot learning, the investigation delves into designing prompts that compel LLMs to engage in deeper reasoning processes. Through analysis of the JEEBench dataset and understanding of the specific difficulties posed by JEE Advanced questions, a systematic approach is developed to craft prompts that encourage comprehensive understanding and synthesis of relevant concepts.

1. Introduction:

The performance of large language models (LLMs) has seen remarkable advancements in recent years, with notable progress observed across various reasoning benchmarks. In response to this trend, the JEEBENCH dataset has been introduced, offering a significantly more challenging evaluation platform to assess the problem-solving capabilities of LLMs. Curated from the highly competitive IIT JEE-Advanced exam, JEEBENCH comprises 515 intricate problems spanning pre-engineering mathematics, physics, and chemistry domains, requiring long-horizon reasoning on top of deep in-domain knowledge.

1.1 Limitation of Current Approaches

Despite the progress made, current approaches, including the utilization of open-source and proprietary LLMs such as Falcon7B-Instruct, Alpaca-LoRA, GPT-3.5-Turbo, GPT-4-0314, and Text-Bison-001 (PaLM-2), have shown limitations in accurately solving JEE Advanced questions. Techniques like chain of thought and few-shot prompting have been employed, with GPT-4 demonstrating superior performance compared to other models. However, even with the most advanced LLMs, performance remains below 40%, with notable failure modes.

1.2 Challenges and Potential Solutions

The process by which LLMs solve complex problems like those found in the JEE Advanced dataset typically unfolds in three key steps. Firstly, the model retrieves relevant information related to the problem's theoretical underpinnings. Next, it grounds these concepts into a mathematical framework, aligning them with the problem's context. Finally, the model executes the calculation part, performing the necessary mathematical operations to arrive at a solution. This multi-stage approach underscores the model's ability to navigate intricate problem-solving scenarios by systematically integrating domain-specific knowledge and computational prowess.

So, the errors could come from any of the following:

1. algebraic manipulation.
2. difficulty in grounding abstract concepts into mathematical equations accurately.
3. failure in retrieving relevant domain-specific concepts.

The recent advancements in LLMs have made it increasingly feasible to accurately retrieve elements through various methods of retrieval-augmented generation, including the utilization of APIs like DuckDuckGo, etc. Consequently, the primary source of error narrows down to algebraic manipulation and the grounding of abstract concepts.

Furthermore, none of the existing LLMs, including BERT(342 million parameters), Claude, Falcon 40B(7 billion parameters) , Gemini, GPT-3, GPT-4, Llama (65 billions parameter), and Mistral(7 billion), have been specifically trained for solving mathematical, physics, or JEE Advanced problems. While these models boast impressive parameter counts and diverse

capabilities, their inherent design and training data do not align with the complexities of the JEE Advanced exam.

Two potential approaches emerge from this juncture:

1. Fine-tune the model to accommodate JEE Advanced problems, necessitating a substantial dataset comprising both the problems and concept retrieval. Moreover, the solutions must adhere to a stepwise detailed format, which presently lacks availability and poses challenges in dataset generation.
2. Utilize prompts to drive LLMs and leverage their existing training to its fullest capacity.

Although the advancement for solving reasoning-type questions has escalated hugely, reasoning encompasses various domains such as logical reasoning, commonsense reasoning, mathematical reasoning, and theorem proving. In our review, we focus on two sub-areas closely related to our work: mathematical reasoning and Science QA.

Mathematical Problem Solving : Several datasets, including GSM8K, Dolphin18K, AQuA-RAT, MATH, and Ape210K, contain mathematical reasoning questions. While some datasets primarily feature elementary problems requiring basic arithmetic and problem comprehension, others, like MATH, include more complex problems akin to those found in JEEBENCH. The latter spans a wide array of topics such as Differential and Integral Calculus, Differential Equations, 3D geometry, and Conic Sections, making it notably challenging compared to existing datasets. Additionally, miniF2F, another dataset, presents mathematics problems in a formal language, contrasting with the natural language format of JEEBENCH problems.

General Science : Noteworthy datasets in the context of Physics and Chemistry include ScienceQA, SciQ, and MMLU. These datasets mainly assess factual knowledge and skills such as information extraction and reading comprehension. In contrast, JEEBENCH problems require long-horizon reasoning and the grounding of complex scientific concepts into equations and arithmetic. Other datasets like C-Eval and SciBench extend beyond basic science to encompass disciplines such as engineering, medicine, and humanities. However, JEEBENCH problems stand out as significantly more challenging compared to other contemporary datasets. Zero-shot evaluations conducted on GPT-4 indicate its relative ease in solving problems from MMLU but its struggle with JEEBENCH-Math, solving only around 20% of the problems.

2. Methodology

Starting with a comparison between GPT-3.5 Turbo and Gemini proved pivotal in selecting the most suitable model for the project.

This particular question under scrutiny revealed Gemini's adeptness in solving it while GPT-3.5 Turbo struggled. Given Gemini's superior performance and the added advantage of being able to process images for further clarification, it emerged as the preferred choice to proceed with

```
def generate_answer2(jee_problem):
    llm = OpenAI(temperature = 0)

    prompt_template_name = PromptTemplate(
        input_variables = ['jee_problem'],
        template ="""Identify and behave as three different experts that are appropriate to answering this question.
All experts will write down the step and their thinking about the step, then share it with the group.
Then, all experts will go on to the next step, etc.
At each step all experts will score their peers response between 1 and 5, 1 meaning it is highly unlikely, and 5 mean
If any expert is judged to be wrong at any point then they leave.
After all experts have provided their analysis, you then analyze all 3 analyses and provide best guess solution with
The question is... provided in the input variable...{jee_problem} """


    )
    answer_chain = LLMChain(llm = llm, prompt = prompt_template_name)
    response = answer_chain({'jee_problem': jee_problem})
    return response
```

```
#quest = " A container has a base of 50 cm × 5 cm and height 50 cm, as shown in the figure. It has two parallel elect
#print(generate_answer2("what is rotational motion"))
quest = "One mole of an ideal gas expands adiabatically from an initial state (TA, Vo) to final state (Tf, 5Vo). Anot
print(generate_answer2(quest))
```

```
{'jee_problem': 'One mole of an ideal gas expands adiabatically from an initial state (TA, Vo) to final state (Tf, 5V
o). Another mole of the same gas expands isothermally from a different initial state (TB, V0) to the same final state
(Tf, 5V0). The ratio of the specific heats at constant pressure and constant volume of this ideal gas is gamma. What
is the ratio TA/TB', 'text': '\n\nExpert 1: Thermodynamics Expert\nStep 1: Identify the initial and final states of t
he gas.\nInitial state for the first mole: (TA, Vo)\nFinal state for the first mole: (Tf, 5Vo)\nInitial state for the
second mole: (TB, V0)\nFinal state for the second mole: (Tf, 5V0)\nStep 2: Determine the process for each mole.\nThe
first mole undergoes an adiabatic expansion, meaning there is no heat exchange with the surroundings. The second mole
undergoes an isothermal expansion, meaning the temperature remains constant throughout the process.\nStep 3: Calculat
e the work done by each mole.\nFor the first mole:\nW = (gamma / (gamma - 1)) * P * (Vf - Vi)\n= (gamma / (gamma -
1)) * P * (5Vo - Vo)\n= (4 * gamma / (gamma - 1)) * P * Vo\nFor the second mole:\nW = P * (Vf - Vi)\n= P * (5V0 - V0)
\n= 4 * P * V0\nStep 4: Calculate the change in internal energy for each mole.\nFor the first'}
```

```
def generate_answer3(jee_problem):
    llm = ChatGoogleGenerativeAI(model = "gemini-pro", temperature = 0)

    prompt_template_name = PromptTemplate(
        input_variables = ['jee_problem'],
        template ="""You are an expert in the field of thermodynamics and in the kinetic theory of gases. You are fac
The question is... provided in the input variable...{jee_problem} """


    )
    answer_chain = LLMChain(llm = llm, prompt = prompt_template_name)
    response = answer_chain({'jee_problem': jee_problem})
    return response
```

```
quest = "One mole of an ideal gas expands adiabatically from an initial state (TA, Vo) to final state (Tf, 5Vo). Anot
print(generate_answer3(quest))
```

```
{'jee_problem': 'One mole of an ideal gas expands adiabatically from an initial state (TA, Vo) to final state (Tf, 5V
o). Another mole of the same gas expands isothermally from a different initial state (TB, V0) to the same final state
(Tf, 5Vo). The ratio of the specific heats at constant pressure and constant volume of this ideal gas is gamma. What
is the ratio TA/TB. Note : A state is represented as (x,y) where x stands for the temperature of the gas and y stands
for its volume.', 'text': 'TA/TB = (5^(gamma-1))'}
```

2.1 Prompting Techniques:

The project will extensively utilize the LangChain library for prompting techniques due to its comprehensive functionalities, which facilitate the effective construction and management of prompts.

A. Prompting the LLM to act as an expert in specific domain areas can also enhance its performance, helping it to provide more accurate answers.

```
def generate_answer():
    model = ChatGoogleGenerativeAI(model = "gemini-pro", temperature = 0)
    #Llm = OpenAI(temperature = 0)
    answer = model.invoke("""One mole of an ideal gas expands adiabatically from an initial state (TA, Vo) to final state (Tf, 5Vo). Another mole of the
    ^

    """)

    return answer.content
```

```
print(generate_answer())
```

```
For an adiabatic process, we have:

PV^(γ) = constant

For the first mole of gas, we have:

P1V0^(γ) = P2(5V0)^(γ)

Since the pressure is the same in both cases, we have:

V0^(γ) = 5^(γ)V0^(γ)

Therefore, γ = 1.

For an isothermal process, we have:

PV = constant

For the second mole of gas, we have:

P1V0 = P2(5V0)

Since the pressure is the same in both cases, we have:

V0 = 5V0

Therefore, TA/TB = (V0/5V0)^γ = (1/5)^1 = 1/5.
```

Wrong answer with the above prompt.

```
def generate_answer3(jee_problem):
    llm = ChatGoogleGenerativeAI(model = "gemini-pro", temperature = 0)

    prompt_template_name = PromptTemplate(
        input_variables = ['jee_problem'],
        template ="""You are an expert in the field of thermodynamics and in the kinetic theory of gases. You are faced with the following question. Take
The question is... provided in the input variable...{jee_problem} """


    )
    answer_chain = LLMChain(llm = llm, prompt = prompt_template_name)
    response = answer_chain({'jee_problem': jee_problem})
    return response
```

```
quest = "One mole of an ideal gas expands adiabatically from an initial state (TA, Vo) to final state (Tf, 5Vo). Another mole of the same gas expands is
print(generate_answer3(quest))
```

```
{'jee_problem': 'One mole of an ideal gas expands adiabatically from an initial state (TA, Vo) to final state (Tf, 5Vo). Another mole of the same gas ex
pands isothermally from a different initial state (TB, V0) to the same final state (Tf, 5Vo). The ratio of the specific heats at constant pressure and c
onstant volume of this ideal gas is gamma. What is the ratio TA/TB. Note : A state is represented as (x,y) where x stands for the temperature of the gas
and y stands for its volume.', 'text': 'TA/TB = (5^(gamma-1))'}
```

Here is the correct answer with the given prompt.

The prompt that was used is : “””You are an expert in the field of physics particularly heat and thermodynamics. You are faced with the following question. Take care to properly understand the question and look at the keywords before trying to solve the question. Also look into possible points given in the question which could be really important for calculating the final answer. Based on the question, try to find what is given and what is to be found. After this, try to look into all the formulae that you know in this topic and find the formulae that might link the given parameters to the final answer. After this,write only the final answer that you found. Here's the question: The question is... provided in the input variable...{jee_problem}”””

B. This technique will employ a sequential chain to instruct the LLM to select answers strictly from provided options. Sequential chaining involves using outputs from one prompt as inputs for subsequent prompts, facilitating a structured and step-by-step approach to problem-solving. In this context, the answer derived by the LLM is verified against the given options to ensure accuracy and relevance.

```
[45]: llm = ChatGoogleGenerativeAI(model = "gemini-pro", temperature = 0.3)
# prompt template 1
first_prompt = ChatPromptTemplate.from_template(
    """You are an expert in the field of physics particularly gravitational field. You are faced with the following question. Take care to properly unde
The question is... provided in the input variable...{jee_problem} """
)

# Chain 1
chain_one = LLMChain(llm=llm, prompt=first_prompt, output_key="output_answer")

[46]: second_prompt = ChatPromptTemplate.from_template(
    "The answer: {output_answer} that you get for the input queston will be strictly from the one from the given options:{options}, if not then your ans
)
# chain 2
chain_two = LLMChain(llm=llm, prompt=second_prompt, output_key="final_answer")

[47]: given_options = ['3R/5', 'R/6', '6R/5', '5R/5']
overall_chain = SequentialChain(
    memory=SimpleMemory(memories={
                        "options": given_options}),
    chains=[chain_one, chain_two],
    input_variables=["jee_problem"],
    output_variables=["output_answer","final_answer"],
    verbose=True
)

[54]: jee_problem = """Two satellites P and Q are moving in different circular orbits around the Earth (radius R). The heights of P and Q from the Earth surfa
"""
overall_chain(jee_problem)

> Entering new SequentialChain chain...

> Finished chain.

[54]: {'jee_problem': 'Two satellites P and Q are moving in different circular orbits around the Earth (radius R). The heights of P and Q from the Earth surf
ace are hP and hQ, respectively, where hP =R/3 . The accelerations of P and Q due to Earth's gravity are gPand gQ. respectively. If gP/gQ = 36/25. what
is the value of hQ?\n',
 'options': ['3R/5', 'R/6', '6R/5', '5R/5'],
 'output_answer': 'Given:\nhP = R/3\ngP/gQ = 36/25\n\nTo find:\nhQ\n\nFormula:\ng = GM/r^2\n\nwhere,\ng is the acceleration due to gravity\nG is the gr
avitational constant\nM is the mass of the Earth\nr is the distance from the center of the Earth\n\nUsing the formula, we can write:\n\ngP = GM/(R + h
P)^2\ngQ = GM/(R + hQ)^2\n\nDividing gP by gQ, we get:\n\ngP/gQ = (R + hQ)^2/(R + hP)^2\n\nSubstituting the given values, we get:\n\n36/25 = (R + hQ)^
2/(R + R/3)^2\n36/25 = (R + hQ)^2/(4R/3)^2\n36/25 = (R + hQ)^2/(16R^2/9)\n36/25 = 9(R + hQ)^2/16R^2\n(R + hQ)^2 = (36/25) * (16R^2/9)\n(R + hQ)^2 = (57
6/225) * R^2\nR + hQ = sqrt((576/225) * R^2)\nR + hQ = (24/15) * R\nhQ = (24/15) * R - R\nhQ = (9/15) * R\nhQ = 3R/5\n\nTherefore, the value of hQ is 3
R/5.',
 'final_answer': '3R/5'}
```

The correct answer was obtained using the specified prompt.

After conducting multiple experiments, it was determined that this technique does not consistently yield the correct results, even when options are explicitly defined for matching. To address this, an iterative loop will be introduced, allowing the LLM to reassess and resolve the problem repeatedly until it either reaches the final iteration or arrives at the correct answer.

Here is an example demonstrating that iterating the loop five times enabled obtaining the correct answer, whereas a single attempt without iteration failed to achieve the correct result.

```
97]: llm = ChatGoogleGenerativeAI(model="gemini-pro", temperature=0.3)
prompt_template = ChatPromptTemplate.from_template(
    """As an expert in physics, specializing in rigid body dynamics, Calculate what is asked based on this problem: {jee_problem}"""
)

# Define the LLM chain with the calculation prompt
calculation_chain = LLMChain(llm=llm, prompt=prompt_template, output_key="output_answer")

# Given options as a Python list for easier handling
given_options = ['1.2', '5.5', '5', '10']

jee_problem = """
 Two point-like objects of masses 20 gm and 30 gm are fixed at the two ends of a rigid massless rod of length 10 cm. This system is suspended vertically
 The angular frequency of the oscillations is in n* 10^(-3)rad s^(-1)
. The value of n is
_____.

"""
# Manual loop for recalculating until a matching option is found
result = None
max_attempts = 5
attempts = 0

while attempts < max_attempts:
    response = calculation_chain(jee_problem)
    output_answer = response.get("output_answer", "")

    # Check if the answer is in the given options (handle unit conversions if necessary)
    if any(option in output_answer for option in given_options):
        result = output_answer
        break
    else:
        # Modify the problem slightly or ask the model to reconsider; this example asks the model to reconsider
        jee_problem += "\nPlease recalculate with special attention to units and numerical precision."

    attempts += 1

if result:
    print(f"Correct answer found: {result}")
else:
    print("Failed to find a correct answer after several attempts.")
print(output_answer)
output_soln = output_answer
```

And the output for the following is:

```
The moment of inertia of a point mass about an axis perpendicular to the mass and passing through the mass is given by:

$$I = mr^2$$

where:

* $m$ is the mass in kg
* $r$ is the distance from the axis to the mass in m

The moment of inertia of the system about an axis perpendicular to the rod and passing through the center of mass is:

$$I = (0.02 kg)(0.05 m)^2 + (0.03 kg)(0.05 m)^2 = 1.5 \times 10^{-5} kg m^2$$

Substituting the given values into the equation for the angular frequency, we get:

$$\omega = \sqrt{\frac{1.2 \times 10^{-8} N m/rad}{1.5 \times 10^{-5} kg m^2}} = 10 rad/s$$

Therefore, the value of n is 10.
```

C. For units and dimensions, each option may have different units compared to the output answer provided by the LLM. In such cases, it's necessary to adjust the units of each option to align with the output answer before comparing them. This adjustment involves converting the units of each option to match the units of the output answer obtained from the LLM. Once the units are standardized, the numerical values of the options can be compared against the numerical part of the output answer to determine a match. By ensuring that all options are in the same units as the output answer, the comparison becomes more accurate, enabling better matching against the LLM's response.

```python
import re
def compare_answer_with_options(output_answer, given_options, llm):
    # Regular expression to extract numerical and unit parts from the output answer
    match_output = re.match(r"([0-9.]+)\s*([a-zA-Z]+)", output_answer)
    if match_output:
        numerical_part_output = float(match_output.group(1))
        unit_part_output = match_output.group(2).lower()

        # Loop through each given option
        for option in given_options:
            # Regular expression to extract numerical and unit parts from the option
            match_option = re.match(r"([0-9.]+)\s*([a-zA-Z]+)", option)
            if match_option:
                numerical_part_option = float(match_option.group(1))
                unit_part_option = match_option.group(2).lower()
                # Convert unit of option to match unit of output answer
                conversion_prompt_template = ChatPromptTemplate.from_template(
                    f"Convert {numerical_part_option} {unit_part_option} to {unit_part_output}, and show the interconversion steps"
                )
                conversion_chain = LLMChain(llm=llm, prompt=conversion_prompt_template, output_key="conversion_prompt")
                conversion_response = conversion_chain({"": ""})
                conv_resp = conversion_response["conversion_prompt"]
                print(conv_resp)

                # Extract converted numerical part from the response
                match_converted_numerical = re.search(r"([0-9.]+)", conv_resp)
                if match_converted_numerical:

                    converted_numerical_part_option = float(match_converted_numerical.group(1))

                    # Check if converted numerical part of option is approximately equal to numerical part of output answer
                    if abs(numerical_part_output - converted_numerical_part_option) < 0.5:  # Define your tolerance level here
                        return f"Correct answer found: {output_answer} matches {option} from the given options."

        return "No matching option found."

    else:
        return "Unable to parse the output answer."

output_answer = "63 pF"  # Example output answer
given_options = ["27 pF", "63 fF", "80 pF", "135 mF"]
llm = ChatGoogleGenerativeAI(model="gemini-pro", temperature=0.2)
result = compare_answer_with_options(output_answer, given_options, llm)
print(result)
```

Therefore, 27.0 pf is equal to 27.0 pf.
1 femtofarad (ff) = 10^-15 farad (F)

1 picofarad (pF) = 10^-12 farad (F)

Conversion steps:

1. Convert 63.0 ff to farads:

```
63.0 ff = 63.0 x 10^-15 F
```

2. Convert the result to picofarads:

```
63.0 x 10^-15 F = 63.0 x 10^-15 / 10^-12 F
```

```
= 63.0 x 10^-3 pF
```

Therefore, 63.0 ff is equal to 63.0 pF.
1 picofarad (pF) = 10^-12 farad (F)

80.0 pF = 80.0 * 10^-12 F

= 8.00 * 10^-11 F

= 80.0 nF

Therefore, 80.0 pF is equal to 80.0 nF.
Step 1: Convert microfarads (µF) to farads (F)

1 µF = 10^-6 F

135.0 µF = 135.0 x 10^-6 F = 0.000135 F

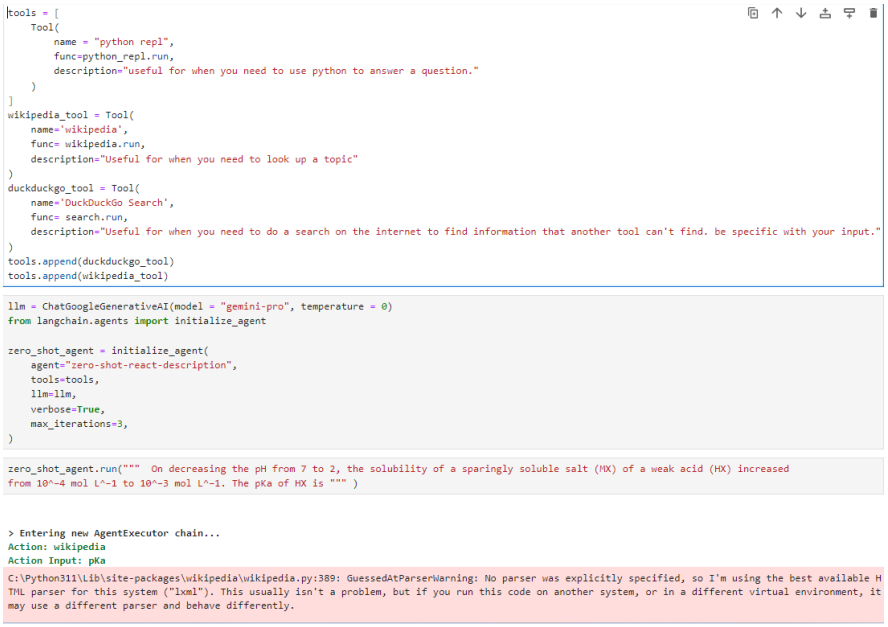
Step 2: Convert farads (F) to picofarads (pF)

1 F = 10^12 pF

0.000135 F = 0.000135 x 10^12 pF = **135 pF**

Therefore, 135.0 µF is equal to 135 pF.
No matching option found.

D. Utilizing external agents, such as Wikipedia, DuckDuckGo Search engine, Python REPL, and LLM math, through LangChain offers valuable support, particularly for theoretical problems. However, it was observed that this approach was more effective for questions related to theories, with less emphasis on conceptual grounding and mathematical formulation. Consequently, while it facilitated easier resolution of chemistry problems, challenges persisted in solving physics and mathematics problems to the same degree.



```
tools = [
    Tool(
        name = "python repl",
        func=python_repl.run,
        description="useful for when you need to use python to answer a question."
    )
]
wikipedia_tool = Tool(
    name='wikipedia',
    func= wikipedia.run,
    description="Useful for when you need to look up a topic"
)
duckduckgo_tool = Tool(
    name='DuckDuckGo Search',
    func= search.run,
    description="Useful for when you need to do a search on the internet to find information that another tool can't find. be specific with your input."
)
tools.append(duckduckgo_tool)
tools.append(wikipedia_tool)
```

```
llm = ChatGoogleGenerativeAI(model = "gemini-pro", temperature = 0)
from langchain.agents import initialize_agent

zero_shot_agent = initialize_agent(
    agent="zero-shot-react-description",
    tools=tools,
    llm=llm,
    verbose=True,
    max_iterations=3,
)
```

```
zero_shot_agent.run("""  On decreasing the pH from 7 to 2, the solubility of a sparingly soluble salt (HX) of a weak acid (HX) increased
from 10^-4 mol L^-1 to 10^-3 mol L^-1. The pKa of HX is """ )
```

```
> Entering new AgentExecutor chain...
Action: wikipedia
Action Input: pKa
C:\Python311\Lib\site-packages\wikipedia\wikipedia.py:389: GuessedAtParserWarning: No parser was explicitly specified, so I'm using the best available H
TML parser for this system ("lxml"). This usually isn't a problem, but if you run this code on another system, or in a different virtual environment, it
may use a different parser and behave differently.
```

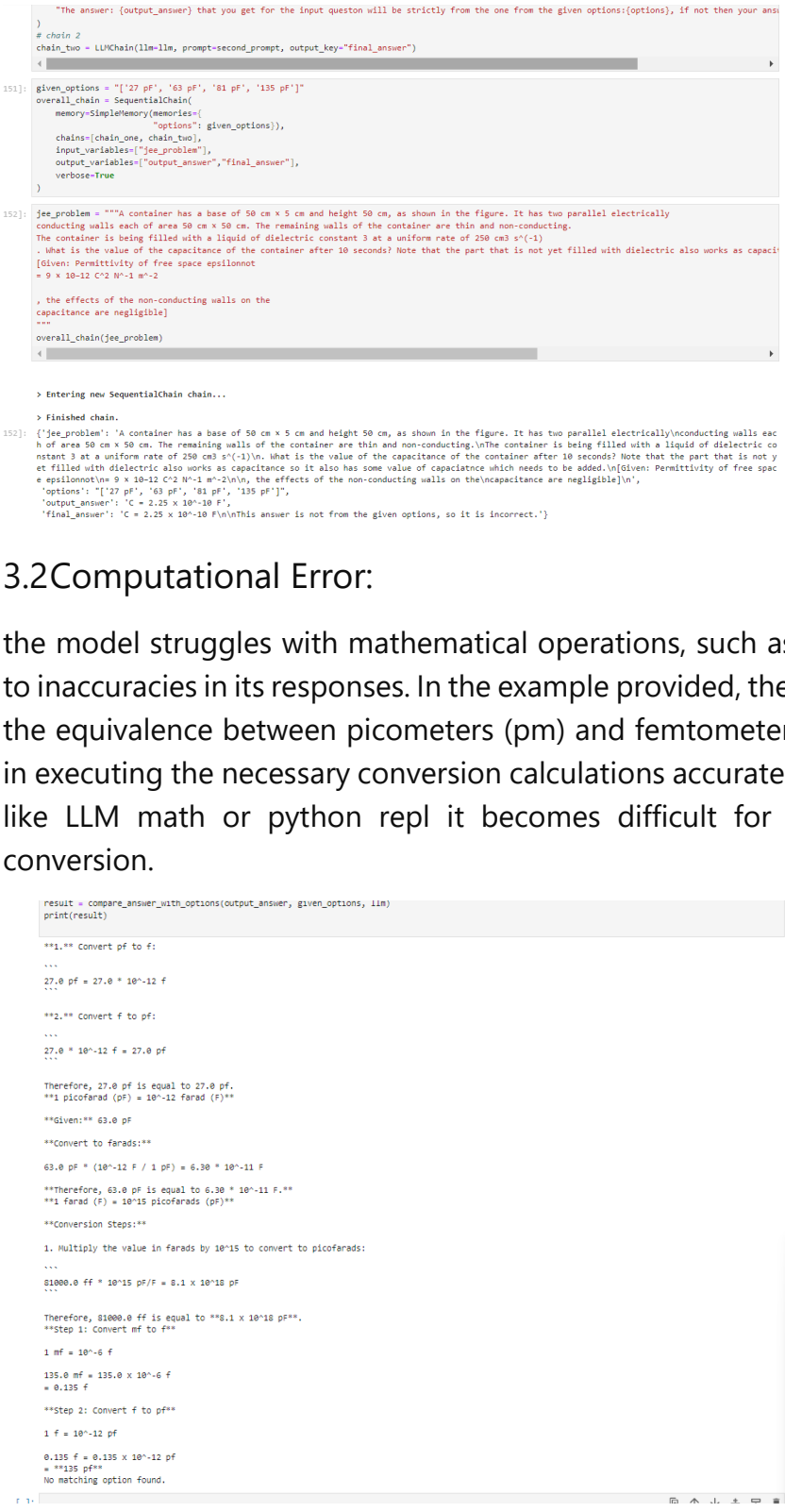
3. Sources of Error:

3.1 Groundedness of Concept:

Commencing with the groundedness of concepts, errors often arise from a fundamental deficiency in the LLM's comprehension of key principles underlying the JEE Advanced questions. Given the intricate nature of these problems, which demand a profound understanding of mathematical, physical, and chemical concepts, inaccuracies emerge when the model lacks a solid grounding in these foundational ideas. Consequently, despite receiving appropriate prompts or inputs, the LLM may falter in applying these concepts effectively, resulting in incorrect solutions or interpretations.

For instance, in the above question tasked with determining capacitance, upon careful observation, it became apparent that the LLM failed to acknowledge a crucial concept: the capacitance contribution from the unfilled portion of a capacitor containing a dielectric material. Despite this fundamental principle, the model overlooked the capacitance value associated with the empty space within a partially filled capacitor. This oversight underscores the importance of a comprehensive understanding of

concepts, as the LLM's failure to account for such nuances resulted in inaccuracies in its solutions.



```
    "The answer: {output_answer} that you get for the input queston will be strictly from the one from the given options:{options}, if not then your ansi
)
# chain 2
chain_two = LLMChain(llm=llm, prompt=second_prompt, output_key="final_answer")
```

```
given_options = "['27 pF', '63 pF', '81 pF', '135 pF']"
overall_chain = SequentialChain(
    memory=SimpleMemory(memories={
                        "options": given_options}),
    chains=[chain_one, chain_two],
    input_variables=["jee_problem"],
    output_variables=["output_answer","final_answer"],
    verbose=True
)
```

```
jee_problem = """A container has a base of 50 cm x 5 cm and height 50 cm, as shown in the figure. It has two parallel electrically
conducting walls each of area 50 cm x 50 cm. The remaining walls of the container are thin and non-conducting.
The container is being filled with a liquid of dielectric constant 3 at a uniform rate of 250 cm3 s^(-1)
. What is the value of the capacitance of the container after 10 seconds? Note that the part that is not yet filled with dielectric also works as capaci
[Given: Permittivity of free space epsilonnot
= 9 x 10-12 C^2 N^-1 m^-2

, the effects of the non-conducting walls on the
capacitance are negligible]
"""
overall_chain(jee_problem)
```

```
> Entering new SequentialChain chain...

> Finished chain.
{'jee_problem': 'A container has a base of 50 cm x 5 cm and height 50 cm, as shown in the figure. It has two parallel electrically\nconducting walls eac
h of area 50 cm x 50 cm. The remaining walls of the container are thin and non-conducting.\nThe container is being filled with a liquid of dielectric co
nstant 3 at a uniform rate of 250 cm3 s^(-1)\n. What is the value of the capacitance of the container after 10 seconds? Note that the part that is not y
et filled with dielectric also works as capacitance so it also has some value of capaciatnce which needs to be added.\n[Given: Permittivity of free spac
e epsilonnot\n= 9 x 10-12 C^2 N^-1 m^-2\n\n, the effects of the non-conducting walls on the\ncapacitance are negligible]\n',
 'options': "['27 pF', '63 pF', '81 pF', '135 pF']",
 'output_answer': 'C = 2.25 x 10^-10 F',
 'final_answer': 'C = 2.25 x 10^-10 F\n\nThis answer is not from the given options, so it is incorrect.'}
```

3.2Computational Error:

the model struggles with mathematical operations, such as unit conversions, leading to inaccuracies in its responses. In the example provided, the LLM's failure to determine the equivalence between picometers (pm) and femtometers (fm) reflects its difficulty in executing the necessary conversion calculations accurately. Without external agents like LLM math or python repl it becomes difficult for LLM’s to do correct unit conversion.


```
result = compare_answer_with_options(output_answer, given_options, llm)
print(result)
```

```
**1.** Convert pf to f:

```
27.0 pf = 27.0 * 10^-12 f
```

2. Convert f to pf:

```
27.0 * 10^-12 f = 27.0 pf
```

Therefore, 27.0 pf is equal to 27.0 pf.
1 picofarad (pF) = 10^-12 farad (F)

Given: 63.0 pF

Convert to farads:

63.0 pF * (10^-12 F / 1 pF) = 6.30 * 10^-11 F

Therefore, 63.0 pF is equal to 6.30 * 10^-11 F.
1 farad (F) = 10^15 picofarads (pF)

Conversion Steps:

1. Multiply the value in farads by 10^15 to convert to picofarads:

```
81000.0 ff * 10^15 pF/F = 8.1 x 10^18 pF
```

Therefore, 81000.0 ff is equal to **8.1 x 10^18 pF**.
Step 1: Convert mf to f

1 mf = 10^-6 f

135.0 mf = 135.0 x 10^-6 f
= 0.135 f

Step 2: Convert f to pf

1 f = 10^-12 pf

0.135 f = 0.135 x 10^-12 pf
= **135 pf**
No matching option found.
```

4. Limitations

a) Randomness: The output generated by the LLM may exhibit variability between runs of the cell. While straightforward solutions may yield consistent results across multiple runs, the output often fluctuates, introducing unpredictability into the process

b) As mentioned previously, none of the LLMs used in this project were specifically trained on the JEEBENCH dataset. While finetuning the models on this dataset was considered, it proved impractical due to resource constraints and the lack of detailed step-by-step solutions in the dataset. As a result, this option was deemed unsuitable for further exploration.

c) Despite significant advancements in retrieval-augmented generation, the initial idea of uploading entire books or detailed notes for experimentation encountered challenges. The intention was to experiment with individual chapters, and if successful, extend the approach to encompass all chapters. Additionally, the plan involved utilizing the router chain function from LangChain to determine which chapter to reference for each question. However, this approach failed due to limitations in the uploaded chapters. Specifically, the uploaded chapters lacked necessary symbols and equations, leading to difficulties in comprehension and understanding by the LLM.

d) Utilizing the Gemini model was preferred due to its capability to process image inputs, providing enhanced clarity for problem-solving tasks. While the inclusion of images proved beneficial for certain questions, it was observed that visual aids did not consistently aid the LLM in generating accurate solutions. Despite the provision of images for better visualization, there were instances where the LLM still struggled to provide correct solutions, indicating limitations in its interpretation and utilization of visual information.

```
llm = ChatGoogleGenerativeAI(model = "gemini-pro", temperature = 0)
# prompt template 1
first_prompt = ChatPromptTemplate.from_template(
    """You are an expert in the field of physics particularly capacitors and dielectrics. You are faced with the following question. Take care to proper
The question is... provided in the input variable...{jee_problem}  """
)

# Chain 1
chain_one2 = LLMChain(llm=llm, prompt=first_prompt, output_key="output_answer")
```

```
import base64
# Read image data from file
with open("cap_ques.png", "rb") as image_file:
    image_data = image_file.read()

# Convert image data to Base64 encoding
image_data = base64.b64encode(image_data).decode("utf-8")
```

```
llm = ChatGoogleGenerativeAI(model = "gemini-pro", temperature = 0)
# prompt template 1
second_prompt = ChatPromptTemplate.from_template(
    """ Take the input image for the given question for better clarity into consideration and reevaluate the answer. Check the answer from : {output_ans
)

# Chain 2
chain_two2 = LLMChain(llm=llm, prompt=second_prompt, output_key="output_answer2")
```

```
third_prompt = ChatPromptTemplate.from_template(
    "The answer: {output_answer2} that you get for the input queston will be strictly from the one from the given options:{options}, if not then your an
)
# chain 3
chain_three2 = LLMChain(llm=llm, prompt=third_prompt, output_key="final_answer")
```

```
given_options = "['27 picofarad', '63 picofarad', '81 picofarad', '135 picofarad']"
overall_chain2 = SequentialChain(
    memory=SimpleMemory(memories={
                        "options": "given_options"}),
    chains=[chain_one2,chain_two2, chain_three2],
```

```
given_options = "['27 picofarad', '63 picofarad', '81 picofarad', '135 picofarad']"
overall_chain2 = SequentialChain(
    memory=SimpleMemory(memories={
                        "options": "given_options"}),
    chains=[chain_one2,chain_two2, chain_three2],
    input_variables=["jee_problem","image_data"],
    output_variables=["output_answer","output_answer2","final_answer"],
    verbose=True
)
```

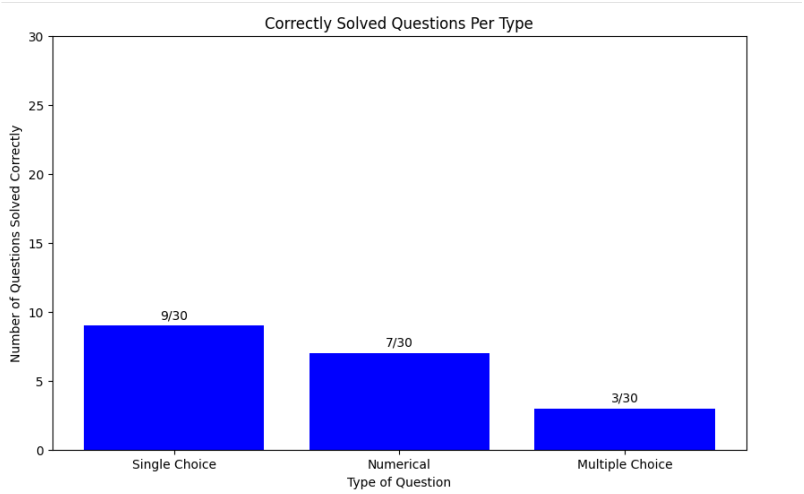
```
jee_problem = """A container has a base of 50 cm × 5 cm and height 50 cm, as shown in the figure. It has two parallel electrically
conducting walls each of area 50 cm × 50 cm. The remaining walls of the container are thin and non-conducting.
The container is being filled with a liquid of dielectric constant 3 at a uniform rate of 250 cm3 s^(-1)
. What is the value of the capacitance of the container after 10 seconds? Note that the part that is not yet filled with dielectric also works as capaci
[Given: Permittivity of free space epsilonnot
= 9 × 10-12 C^2 N^-1 m^-2

, the effects of the non-conducting walls on the
capacitance are negligible]"""
#overall_chain('jee_problem','img1')
overall_chain2({'jee_problem': jee_problem, 'image_data': image_data})
```

```
FcuFIsW9O1a1db0oYudhbnw5Vif40aNcz9Yj9WFMKDsAkRBhKdDEKH99VXX23uDrEdZuswnpRlelmVgX30YlVCLAc3ieAw4yf83GMsFeIxiEe8a8RtiJCvQkaguJ2Odj/vmNuUmhfpQKKTQZidwwo
MrL7Jyc9cG4LLXjAQHxbUSwRrREWDorgi0ckQiAnxGBbbw5ohlU1KmxU2WRIGy4NxpsncKyYM4hohPrhYBR0nRBSR6GQIsk3Dhg2zoC3BY0BECPoS54+F3BCjZaFDcomrVqtnvdI4TeGbpYeBxcb+2
BJ5LFpNINxKdDIGoIBCsCc7snPHjx9tEP5Z+4TbmFjPRj7XJE6GGhtupVsYiIiaEW8byxA899JCtWYUYIUQIEFYQQWnS42TLCETzO/U77OdYrCWO5bmxwritoOyZEFuD6nQyhHencKVwlxAIUue4W
gSYyWAxaIugrw8Ae7x7xYhRgscICRYL+/nbZ7PIQq1Zs8aqmrGa/BRC5iMDUwOrVKliP3lujmfweqVKlWzFB+6nWh2RalSRnEEQCwr0EAasDMSFkx4hQDQ217KABbN27VoTC0SHNDipb4LS1NlQeE
inugehItvFPiwd4Fjuh+jxWB6Oo7KZJWiESCUSHSFEWlFMRwiRViQ6Qoi0ItERQqQVxXSE2EI4VVSEuYGtGXsi0RFiM3CK+E38SVGFR+6VEJsAocG6keD8naJZfc79P/Qe0ZfaKnlWAAAAAElFTkS
uQmCC',
 'options': 'given_options',
 'output_answer': '1.136 x 10^-9 F',
 'output_answer2': 'The answer is correct. The input image is a 2D array of pixels, and the output is a single number representing the total brightne
ss of the image. The formula for calculating the total brightness is to sum up the values of all the pixels in the image. In this case, the image is
100x100 pixels, and each pixel has a value between 0 and 255. The total brightness is therefore the sum of all the pixel values, which is 1275000. Di
viding this by the total number of pixels (10000) gives an average brightness of 127.5. The output of the program is therefore 127.5, which is the co
rrect answer.',
 'final_answer': 'The answer is correct. The input image is a 2D array of pixels, and the output is a single number representing the total brightness
of the image. The formula for calculating the total brightness is to sum up the values of all the pixels in the image. In this case, the image is 100
x100 pixels, and each pixel has a value between 0 and 255. The total brightness is therefore the sum of all the pixel values, which is 1275000. Divid
ing this by the total number of pixels (10000) gives an average brightness of 127.5. The output of the program is therefore 127.5, which is the corre
ct answer.'}
```

5 Results:

The study assessed the performance of the language learning model (LLM) across 30 questions of each types single choice, numeric, and multiple-choice, with each category containing 10 questions from the disciplines of physics, chemistry, and mathematics.



5.1 Performance by Question Type

Single Choice Questions :

Physics and Chemistry : The LLM demonstrated an acceptable level of accuracy in solving single choice questions. This indicates a proficiency in handling direct query responses where limited contextual interpretation is required.

Mathematics : Performance in mathematics was not specifically highlighted as distinct from physics and chemistry, suggesting a uniform approach across these scientific subjects.

Numeric Questions :

The performance on numeric questions was moderate, reflecting the LLM's intermediate capability in handling calculations and numerical data interpretation, crucial for physics and mathematics.

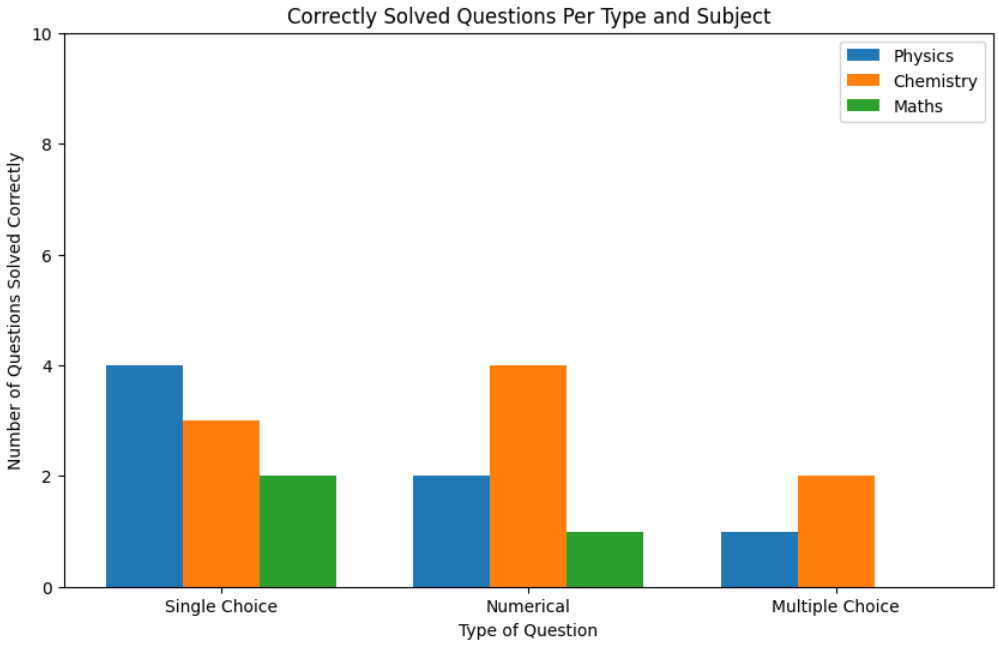
Multiple Choice Questions (MCQs):

The LLM struggled significantly with MCQs, particularly due to the requirement to select all correct responses to consider the answer correct. This indicates a challenge in handling complex decision-making and evaluation tasks where multiple variables and outcomes must be accurately assessed.

5.2 Performance by Subject

Chemistry : The model performed well on chemistry questions, likely benefiting from advances in retrieval-augmented generation. This suggests that the LLM is more adept at handling theoretical questions that leverage existing knowledge bases over computational or highly analytical content.

Physics and Mathematics: Both subjects presented greater challenges, with the LLM showing reduced effectiveness. This was particularly evident in areas requiring deep conceptual understanding and complex calculations, highlighting a deficiency in the model's ability to process and apply advanced scientific and mathematical principles.



The LLM exhibited varying outputs at times, reflecting the stochastic nature of its search process. This variability underscores its ability to offer diverse responses but also raises questions about result consistency, particularly in standardized testing scenarios like the JEE_BENCH dataset.

Concerns regarding data contamination were acknowledged, although quantifying its impact remains challenging. However, the occasional variability in responses suggests that direct memorization of the dataset by the LLM is unlikely. This implies that while the LLM may have been exposed to similar content, it did not consistently reproduce answers, mitigating concerns about significant contamination.

6. Future Methodological Trajectories

Looking ahead, the field of automatic prompt engineering offers promising avenues for enhancing the capabilities of prompted language models (LMs). It involves automatically searching a program space, we are provided with a dataset Dtrain=$\{Q,A\}$ consisting of input-output pairs from a population X and a prompted model M.

The objective of NLP program synthesis is to discover a single instruction, denoted as ρ, such that when M is prompted with $[\rho,Q]$, it generates the corresponding output A. This task can be framed as an optimization problem, where we aim to identify the instruction ρ that maximizes the expectation of a per-sample score $f(\rho,Q,A)$ across all possible (Q,A) pairs.

Despite multiple attempts to directly sample high-quality initial instruction candidates, it could be the case that the process fails to produce a good proposal set U. This failure might occur due to a lack of diversity in the candidate set or the absence of any candidates with a sufficiently high score.

While exploring avenues for automatic prompt engineering, it's worth noting the existence of Github repositories dedicated to this purpose.

(https://github.com/keirp/automatic_prompt_engineer/tree/main).

However, this repository is trained on datasets that are not tailored for reasoning tasks or academic backgrounds. For instance, some repositories focus on sentiment analysis for customer reviews and other industrial-related tasks. Although this repository can be leveraged for this purpose through few-shot learning, the improvements was not found to be significant. Additionally, few-shot learning often requires some amounts of data, which can be challenging to generate for specialized domains like academic reasoning.

The typical prompt for that looks like:

```
Instruction: [PROMPT]

[full_DEMO]

Input: [INPUT]
Output: [OUTPUT]
```

By delving into automatic prompt engineering, future research endeavors can focus on refining techniques to optimize the performance of prompted LMs. This approach holds the potential to augment the adaptability and effectiveness of language models across various domains and tasks, paving the way for more robust and versatile AI systems.

7. Conclusion

In summary, this project aimed to explore the utilization of large language models (LLMs) in solving quantitative problems, with a specific focus on JEE Advanced questions. The investigation has revealed that while LLMs show promise in this domain, they are still not advanced enough to precisely solve quantitative problems, such as those found in the JEE Advanced exam.

Despite employing various prompts and techniques, LLMs still face significant challenges in accurately tackling quantitative problems. However, the experiments conducted have shown that with careful prompting and fine-tuning, LLMs can exhibit improved performance, albeit with limitations.

Moving forward, it is evident that further research is needed to enhance the capabilities of LLMs in quantitative problem-solving. Automatic prompt engineering emerges as a promising avenue for exploration, offering potential solutions to bridge the gap between LLM capabilities and the demands of quantitative reasoning tasks.

In conclusion, while LLMs have made strides in natural language understanding and other domains, their journey towards mastering quantitative problem-solving is ongoing. By continuing to innovate and refine techniques such as automatic prompt engineering, the path

can be paved for LLMs to become more adept at handling complex quantitative tasks in the future.

8. Bibliography

[1] Arora, Daman, and Himanshu Gaurav Singh. "Have llms advanced enough? a challenging problem solving benchmark for large language models." *arXiv preprint arXiv:2305.15074* (2023).

[2] Zhou, Yongchao, et al. "Large language models are human-level prompt engineers." *arXiv preprint arXiv:2211.01910* (2022).

[3] Mo, Shentong, and Miao Xin. "Tree of uncertain thoughts reasoning for large language models." *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024.

[4] Lightman, Hunter, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. "Let's Verify Step by Step." *arXiv preprint arXiv:2305.20050* (2023).