UCCD3074 Deep Learning for Data Science
Group Assignment
Multiple-Choice Science Question Answering with Large Language Models

Research-based [ ] Application-based [X]

Name	Ng Jing Ying (Leader)	Tan Chun Kit	Teoh Zhen Quan	Choong Wei Theng
Programme	CS	CS	CS	CS
ID	21ACB01845	20ACB02818	21ACB01805	20ACB02727
Contribution	1/4	1/4	1/4	1/4

1. INTRODUCTION

The rise of adopting transformer architectures to develop Language Models has been growing fast in recent years due to their superior language reasoning capabilities compared to traditional RNN-based and LSTM models. Since the emergence of ChatGPT, it has been more and more open-source Large Language Models (LLMs) available to the public. These models can do various tasks requiring logical reasoning.

To fully explore the capabilities of LLMs, equipping them with scientific reasoning skills is crucial because many real-world classification tasks require a strong understanding of scientific knowledge. For example, sentiment analysis, a well-known problem in Natural Language Processing (NLP) falls under the domain of language science.

In this project, we aim to explore the scientific reasoning capabilities of recent large language models by fine-tuning existing models. The problem statement is stated as follows:

(1) Existing text generation LLMs are inefficient.

While LLMs establish commendable context understanding capabilities, they are prone to generating inconsistent output due to their stochastic nature. There are two aspects regarding this negative implication. Firstly, the model may struggle to give consistent answers when facing challenging domain-specific questions [1]. To improve model reasoning and consistency, chain-of-thought (CoT) prompting [2] was introduced, leading the model to explain step-by-step before making the final prediction. While literature [1] showed significant improvement in applying CoT on scientific multiple-choice questions (MCQs), CoT indeed reduces the time efficiency, as the model needs to generate more output tokens compared to without using CoT. On the other hand, while the approach from [3] allowed models to use tools such as internet search and text queries to improve reasoning capabilities, such an approach may not be efficient all the time.

Secondly, the autoregressive nature of text generation models, which is "predicting the next possible token" challenges developers to integrate LLMs into the application system due to the inconsistent format generated. For instance, for multiple choice questions, the model may output "The answer is A", even though developers strive to force the model to output "A" as the first output token in the prompt [4].

Pretrained LLMs are often fine-tuned to perform downstream tasks. For instance, the last fully connected layer of an LLM can be replaced with a linear layer with 6 neurons for an emotion multiclass classification problem. While this approach addresses format inconsistency issues,

such models will lose pre-trained representations after finetuning, and can no longer be generalized to other tasks. From the application developers' point of view, they need a model that is both flexible and able to provide consistent outputs. As such, we intend to fine-tune an instruction-tuned model so that it can reach both expectations.

(2) Existing LLMs are commendable, however, they are difficult to fine-tune

While state-of-the-art LLMs show commendable reasoning capabilities, having billions of parameter sizes makes them difficult to fine-tune. For instance, GPT-3 [5] contains 175 billion parameters. Even though other SOTA models such as LLaMA and Mistral have as low as 7 billion parameter size, such models can hardly be fine-tuned using consumer GPUs. To address this, techniques such as Low-Rank Adaptation (LoRA) [6] have been developed to enable finetuning models using reduced GPU memory and compute power. Hence, we intend to discover the most efficient fine-tuning approach for an open-source LLM, which is Mistral 7B instruct v0.2.

In short, the project has two objectives:

(1) To fine-tune an instruction-tuned LLM in solving scientific MCQs.
The fine-tuned model is expected to predict the answer label based on the given choices in a question. Developers can just prompt the problem into the model and retrieve the answer without any formatting issues.
(2) To implement the LoRA method during the fine-tuning process.
We show that our approach can be conducted using consumer GPUs.

The contribution of our study is as follows:

(1) Tested against knowledge-intensive MCQs: ScienceQA, our MCQA model is on par with SOTA LLMs, such as GPT-3 and ChatGPT.
(2) To the best of our knowledge, most of the proprietary instruction-tuned LLM providers are text-generation models and not customized for multiple-choice classification tasks. Instead, our models allow users to "prompt", just like using these LLMs and get reliable output labels, ranging from 0 to 4.
(3) The entire finetuning process takes around 6 hours with GPU RAM usage not exceeding 16GB, allowing future researchers to replicate the training pipeline using consumer GPUs.

2. SYSTEM DESIGN

The high-level view of the entire finetuning process is shown in Figure 2.1, encompassing data preprocessing, model training and validation, hyperparameter tuning and final benchmarking.

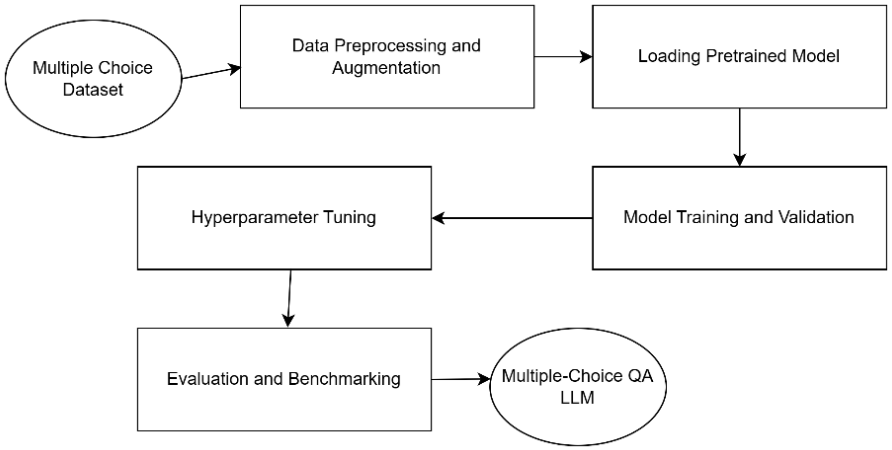

Figure 2.1 System Design

2.1 Data Preprocessing and Augmentation

The chosen dataset is ScienceQA [1] which comprises 21,208 samples, with 54% of Natural Science, 25% Language Science and 21% Social Science questions. 48% of the samples contain image context, while 30.8% of the samples contain both image and text context.
Firstly, we split the dataset into training (12.7k samples), validation (4.24k samples) and testing (4.24k samples) sets according to Hugging Face settings. However, we only use 40% of the training set for better training effectiveness. Since we are not building visual models, we use GPT-2 Vision Transformer (ViT) to generate image context for each image, which refers to the literature [1] method. Then, we extract key attributes like subject type, question, context and choices that served as input, and answer that served as label.
Now, we define the prompt template to fit in the samples. We align the prompt according to Mistral's official guideline [7], which is to put [INST] to highlight the instruction part and use a delimiter such as "##" to let the model know the text boundary. Figure 2.2 shows the prompt template for all the samples:

```
[INST] You are a {subject type} {role}. {instruction}
## Question:
{question}
## Hint:
{text context}
{image context}
## Choices:
{choices} [/INST]
## Answer:
```

Figure 2.2 Prompt Template

As shown in the figure, with the "Answer:" prefix, the model is expected to output the label of the answer. To ensure that our model does not overfit a specific instruction template, a certain degree of randomization is applied, details of each parameter are as follows:

(1) The model will be hinted by its expertise, such as natural science, language science or social science. Then, roles such as "expert", "teacher", and "lecturer" are randomly inserted into the prompt.

(2) As described in the literature [8], we want our model to be unbiased to specific choices. Hence, choices are shuffled, and we ensure that the entire training set is balanced. To "pad" the questions that do not have 5 options, we add options "none of these", "all of these" and "I don't know" consecutively.

(3) Instruction. We prepare three different kinds of instructions. Besides the regularization effect, this approach can also ensure that the model will not be too biased to specific instructions and can be more flexible in production.

Unfortunately, we notice that GPT-2 ViT struggles to generate good image context most of the time. To prevent the model learning from being affected by irrelevant knowledge, we add a "potentially misleading" alert into the instruction, so that the model may learn to pay less attention to irrelevant information. This can also help the model in distinguishing irrelevant information in production mode.

2.2 Model loading

The chosen pretrained model is Mistral v0.2 Instruct with 7 billion parameters, it is a Causal Language Model. As shown in Figure 2.2, the model starts with calculating embeddings at the first layer, followed by 32 decoder layers. Then, it is overridden as a sequence classification model by replacing the last layer with a new linear layer of 5 neurons. From now on, we name this model as Mistral Multiple-Choice Question Answering (MCQA) Model. Previously, this layer was used to output the next possible token, and the output has a size of 32, 000. Now the model is designed to output one of five possible labels, ensuring output consistency.

```
MistralForSequenceClassification(
  (model): MistralModel(
    (embed_tokens): Embedding(32000, 4096)
    (layers): ModuleList(
      (0-31): 32 x MistralDecoderLayer(
        (self_attn): MistralSdpaAttention(
          (q_proj): Linear4bit(in_features=4096, out_features=4096, bias=False)
          (k_proj): Linear4bit(in_features=4096, out_features=1024, bias=False)
          (v_proj): Linear4bit(in_features=4096, out_features=1024, bias=False)
          (o_proj): Linear4bit(in_features=4096, out_features=4096, bias=False)
          (rotary_emb): MistralRotaryEmbedding()
        )
        (mlp): MistralMLP(
          (gate_proj): Linear4bit(in_features=4096, out_features=14336, bias=False)
          (up_proj): Linear4bit(in_features=4096, out_features=14336, bias=False)
          (down_proj): Linear4bit(in_features=14336, out_features=4096, bias=False)
          (act_fn): SiLU()
        )
        (input_layernorm): MistralRMSNorm()
        (post_attention_layernorm): MistralRMSNorm()
      )
    )
    (norm): MistralRMSNorm()
  )
  (score): Linear(in_features=4096, out_features=5, bias=False)
)
```

Figure 2.2 Model Architecture

Then, the model is wrapped using the LoRA model. LoRA is then applied to all attention blocks, which are q_proj, k_proj, v_proj and o_proj as shown in Figure 2.3. All original parameters remain frozen, and only the newly added layers will be trained. This accelerates the training process while ensuring memory efficiency.

```
(layers): ModuleList(
  (0-31): 32 x MistralDecoderLayer(
    (self_attn): MistralSdpaAttention(
      (q_proj): lora.Linear4bit(
        (base_layer): Linear4bit(in_features=4096, out_features=4096, bias=False)
        (lora_dropout): ModuleDict(
          (default): Dropout(p=0.1, inplace=False)
        )
        (lora_A): ModuleDict(
          (default): Linear(in_features=4096, out_features=64, bias=False)
        )
        (lora_B): ModuleDict(
          (default): Linear(in_features=64, out_features=4096, bias=False)
        )
        (lora_embedding_A): ParameterDict()
        (lora_embedding_B): ParameterDict()
      )
      (k_proj): lora.Linear4bit(
        (base_layer): Linear4bit(in_features=4096, out_features=1024, bias=False)
        (lora_dropout): ModuleDict(
```

Figure 2.3 Model Architecture after wrapped with LoRA Model

2.3 Model training and validation

As for our project objective, we restrict our GPU memory consumption to 16GB. Thus, we only choose the sample that has an input sequence length of less than 500. As a larger batch size will slow down the training and lower accumulation steps will increase the GPU memory consumption, we set the batch size to 16 and gradient accumulation steps to 4, keeping the product of them in 64. The optimizers are the variations of Adam, which will be experimented with. We enable FP16, also known as mixed precision training, so that some of the parameters will be working in 16-bit precision, effectively reducing memory usage. Lastly, our training steps are restricted to 200. Hyperparameter tuning details are explained in Chapter 3.

2.4 Evaluation and Benchmarking

After the best MCQA model is obtained, we proceed to the evaluation phase. Our MCQA model will be compared with state-of-the-art models such as GPT-3 and Unified QA using answer accuracy as the metric. As an ablation study, the baseline model, which is Mistral v0.2 instruct without fine-tuning, and Mistral v0.2 instruct fine-tuned for CoT will be compared with our Mistral MCQA model.
It is imperative to note that when evaluating causal language models like Mistral instruct and Mistral fine-tuned for CoT, they occasionally output tokens that are different from the alphabetical labels despite providing prefixes in the prompt. For benchmarking purposes, we take the first generated token right after the "Answer:" prefix. In case the model does not output the answer label properly, a heuristic correction mechanism will be applied. This post-processing strategy mimics typical developer behaviour when using LLMs in production. On the other hand, as Mistral MCQA consistently outputs one out of five labels, no post-processing step is needed.

3. EXPERIMENT & EVALUATION

3.1 Evaluation Metrics.

The ScienceQA validation set is used during training, and the testing set will be used as the evaluation dataset in this experiment. Accuracy metrics will be used to determine the reliability of the model in a multi-class classification problem. A prompt will be given to the LLM model and either option A, B, C, D, E, or Invalid is expected to be the output. The evaluated categories are natural science, social science, language science, text context, image context, and no context. Text context refers to the hint attribute provided in the ScienceQA dataset, which describes samples' additional information to answer the question. Image context is extracted from the image given in the dataset and passed to the model as input since the focus of the experiment is not VLM. No context refers to either category mentioned without text and image context. These evaluation strategies align with the literature [1] [3] to ensure a fair comparison.

3.2 Implementation details.

We apply the hyperparameter settings mentioned in 2.3 and experiment with the difference between 8-bit BNB and paged AdamW 32-bit optimizer when fine-tuning Mistral for MCQA. Then, we apply the same hyperparameter settings to fine-tune Mistral for CoT as an ablation study. Since fine-tuning models for CoT generally requires much more GPU memory, we use GPU A100 in Google Colab specifically for that model. For other experiments, we performed on GPU P100 in Kaggle which has a GPU RAM limit of 16GB.

3.3 Hyperparameter tuning

Both experiments below take approximately 6 hours including the validation process. After 200 steps, we observed that an 8-bit quantized Bits-and-Bytes (BNB) optimizer with a dynamic learning rate has the best training and validation loss. It has steadier training and validation loss compared to Paged AdamW 32-bit optimizer. Hence, the model that is fine-tuned using BNB optimizer is selected.

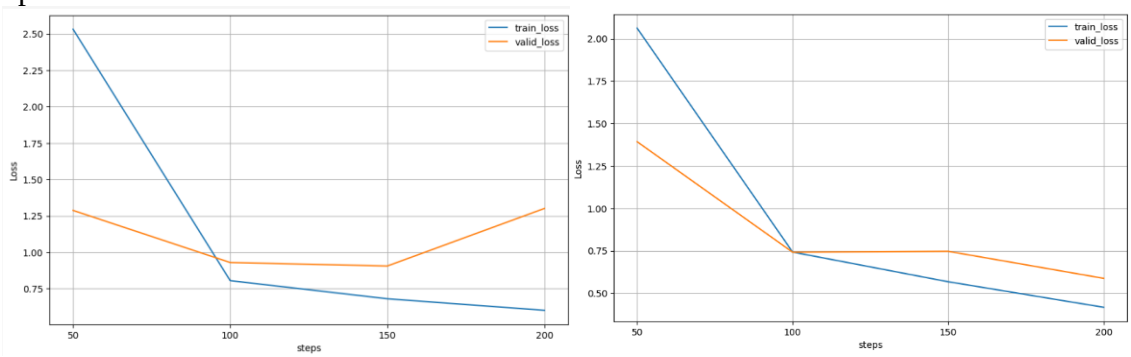

Figure 3.1 Paged AdamW 32-bit optimizer (Left), 8-bit BNB optimizer (Right)

Optimizer	Learning rate	Training loss	Validation loss	Validation accuracy
Paged AdamW 32-bit	1e-4	0.6016	1.301	0.7524
8-bit Quantized BNB	dynamic	0.4166	0.5872	0.7868

Table 3.1 Comparison between Paged AdamW 32-bit Optimizer and 8-bit BNB Optimizer

3.4 Benchmarking Results

Here is the overview of the result compared to other SOTA models.

3.4.1 Result against ScienceQA Datasets

Table 3.2 shows the accuracy scores of different models against the ScienceQA dataset.

(i) **UnifiedQA.** Literature [1] shows that finetuning UnifiedQA to perform with and without CoT has results of 74.11 and 70.12 respectively. This indicates that CoT only pushes the model performance of 3.99%.

(ii) **GPT-3.** GPT-3 without finetuning can reach an accuracy of 74.04% even in zero-shot settings. The model accuracy increased to 75.17% if GPT-3 is guided to perform CoT.

(iii) **ChatGPT**. Literature [3] experiment shows that ChatGPT (GPT-3.5) reaches 78.31% accuracy. After integrating their methods with tools, the accuracy reaches 79.93%, which is the highest in the table below.

(iv) **Ablation studies**. While experimenting with the baseline Mistral, the accuracy is 70.79%, which is like the UnifiedQA without CoT. Interestingly, fine-tuned Mistral for CoT has lower accuracy in general compared to the original. This is reasonable as the model needs to learn harder to perform CoT to output proper explanation and format. Hence, accuracy is expected to be improved if more training steps are set. Nevertheless, the hyperparameters are almost the same as training our model, this proves the inefficiency of finetuning CoT models. As for the fine-tuned Mistral model for MCQA, it reaches an average accuracy of 79.72%. which is an +8.93% improvement compared to the baseline model. This accuracy is close to the best model, Chameleon ChatGPT as shown in the table. The model wins for the text context and image context, with other categories right behind the best model. This shows that finetuning an instruct model with proper instruction will have significant improvement compared to other approaches. This improvement is amplified by the fact that our model has better time efficiency since no CoT and tools are applied.

Model	#Size	#tuned	NAT	SOC	LAN	TXT	IMG	NO	Avg
Random [1]	-	-	40.28	46.13	29.25	47.45	40.08	33.66	39.83
Human [1]	-	-	90.23	84.97	87.48	89.60	87.50	88.10	88.40
UnifiedQA [1]	223M	223M	70.12	68.16	74.91	63.78	61.38	77.84	70.12
UnifiedQA CoT [1]	223M	223M	71.00	**76.04**	78.91	66.42	66.53	81.81	74.11
GPT-3 (0-shot) [1]	175B	0M	75.04	66.59	78.00	74.24	65.74	79.58	74.04
GPT-3 CoT [1]	173B	0M	75.44	70.87	78.09	74.68	67.43	79.93	75.17
ChatGPT CoT [3]	175B	0M	78.82	70.98	83.18	77.37	67.92	86.13	78.31
Chameleon (ChatGPT) [3]	175B	0M	**81.62**	70.64	**84.00**	79.77	70.80	**86.62**	**79.93**
Mistral v0.2 Instruct	7B	0M	72.78	65.92	70.64	71.80	63.96	73.52	70.79
Fine-tuned Mistral CoT	7B	55M	68.83	69.97	71.18	69.21	63.96	71.36	69.68
Fine-tuned Mistral MCQA	7B	55M	80.81	74.69	81.54	**80.06**	**71.89**	83.97	79.72

Table 3.2: Evaluation of different models on different categories in accuracy (%).
#Size = parameter size, #tuned = number of tuned parameters; NAT = natural science, SOC = social science, LAN = language science, TXT = text context, IMG = image context, NO = no context

3.5 Error Analysis

There are two kinds of errors to be discussed.

3.5.1 MCQA Model does not update the factual knowledge after finetuning

Firstly, some error is caused by the wrong knowledge in the baseline model. As shown in Figure 3.2, a natural science question, the MCQA model predicts the answer 'rr'. Prompting the same question to Mistral v0.2 instruct using similar templates, Mistral baseline gives the same answer with a wrong explanation. This shows that the model did not manage to learn this fact even after finetuning.

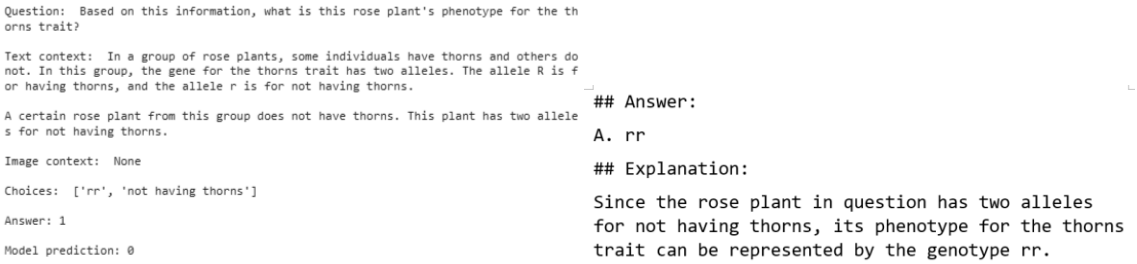

Figure 3.2 Same Question given to MCQA (left side) and Baseline Mistral (right side)

3.5.2 Limitation of Image Captioning Model

From Figure 3.3, it shows that the image contains words such as "FF" or "Ff", however, the GPT-2 captioning model caption the image wrongly (there is no any clock or street sign). Hence, this challenges our MCQA model to provide the correct answer.

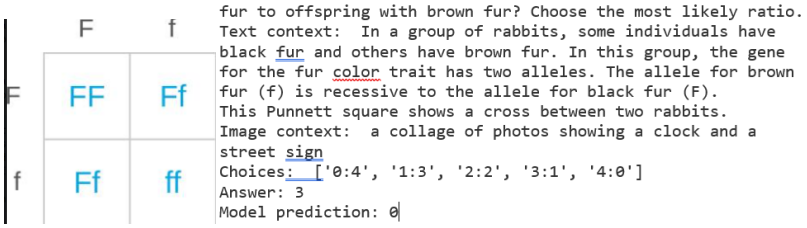

Figure 3.3 Left Side shows the Input Image, Right Side shows the Question with Model Output

5. CONCLUSION

In conclusion, we have shown that:

(1) Without compromising reasoning capabilities, we can fine-tune an LLM as a multiclass classification model to predict the possible answers to ensure consistent output.
(2) As this approach outperforms most state-of-the-art models which include CoT or tools integration, we show that our approach can achieve similar performance with minimal inference time.
(3) It is possible to apply LoRA techniques to perform fine-tuning using our approach. This proves that our proposed approach is efficient.

As for the future work, we plan to:

(1) Currently, our model has limited capability due to the inability of the image captioning model to caption text properly. In the future, we shall integrate a vision encoder to fully explore this training framework.
(2) We shall test on a GPU with more RAM and computation power to see if the model's performance can be further improved.

REFERENCES

[1] P. Lu et al., "Learn to Explain: Multimodal Reasoning via Thought Chains for Science Question Answering," *arXiv.org*, Oct. 17, 2022. https://arxiv.org/abs/2209.09513.

[2] Jason Zhanshun Wei et al., "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," Jan. 2022, https://doi.org/10.48550/arxiv.2201.11903.

[3] P. Lu et al., "Chameleon: Plug-and-Play Compositional Reasoning with Large Language Models," *arXiv.org*, May 24, 2023. https://arxiv.org/abs/2304.09842.

[4] X. Wang et al., "'My Answer is C': First-Token Probabilities Do Not Match Text Answers in Instruction-Tuned Language Models," *arXiv.org*, Feb. 2024, https://doi.org/10.48550/arxiv.2402.14499.

[5] T. Brown et al., "Language Models are Few-Shot Learners," *arXiv.org*, May 2020, https://doi.org/10.48550/arxiv.2005.14165.

[6] E. J. Hu et al., "LoRA: Low-Rank Adaptation of Large Language Models," *arXiv.org*, Jan. 2021, https://doi.org/10.48550/arxiv.2106.09685.

[7] "Introduction | Mistral AI Large Language Models," *docs.mistral.ai*. https://docs.mistral.ai/.

[8] C. Zheng, H. Zhou, F. Meng, J. Zhou, and M. Huang, "Large Language Models Are Not Robust Multiple Choice Selectors," *arXiv.org*, Feb. 21, 2024. https://arxiv.org/abs/2309.03882.

[9] "Efficient training on a single GPU," Efficient Training on a Single GPU, https://huggingface.co/docs/transformers/v4.20.1/en/perf_train_gpu_one (accessed Apr. 21, 2024).

[10] "Nlpconnect/VIT-GPT2-image-captioning · hugging face," nlpconnect/vit-gpt2-image-captioning · Hugging Face, https://huggingface.co/nlpconnect/vit-gpt2-image-captioning (accessed Apr. 21, 2024).