STA5007 Assignment 3

Li Xuanran
12312110

December 11, 2025

1 Q1

Explain and answer the following questions in your own words. You may refer to lecture notes for more details.

1.1

(2 pts) Briefly describe the **goals**, **gained abilities**, and **main challenges** of **Pretraining**, **Instruction Tuning**, and **Alignment Tuning**. Give representative models for each stage.

Solution.

For **Pretraining**,

- Goal: To teach the model the rules of languages and to acquire general knowledge from the text data, resulting in a strong foundation model.
- Gained ability: It learns to predict the next word or token, understand and generate text and code. It also has the ability of in-context learning, reasoning and doing CoT.
- Main challenge: Get sufficient large data scale and good data quantity. Large-scale distributed training, massive computational resources required, and lack of training stability.
- Representative model: GPT-3, LLaMA, PaLM.

For **Instruction Tuning**,

- Goal: To enhance the model's ability to follow diverse and complex human instructions and formats.
- Gained ability: Instruction following, multi-task generalization, and complex reasoning.
- Main challenge: Large-scale diverse instruction data, and the difficulty of improving the code and reasoning abilities.
- Representative model: InstructGPT, FLAN-T5.

For **Alignment Tuning**,

- Goal: To align the output with human values, safety constraints and preferences.

- Gained ability: Generate honesty, safety and unbiased response, and reject instructions that violate human values or beyond its ability.
- Main challenge: Accurately defining and quantifying human preference, which can be subjective, the high cost of human annotation, and with SFT or RLHF.
- Representative model: ChatGPT, Claude, Sparrow.

□

1.2

(1 pt) Read the blog A Closer Look at Large Language Models Emergent Abilities and explain the emergent abilities with various examples.

Solution.

1. Emergent abilities only exist in large models. From Figure 1, we find that then model performance does not increase much when the model size is relatively small, but significantly increase when the model becomes large.

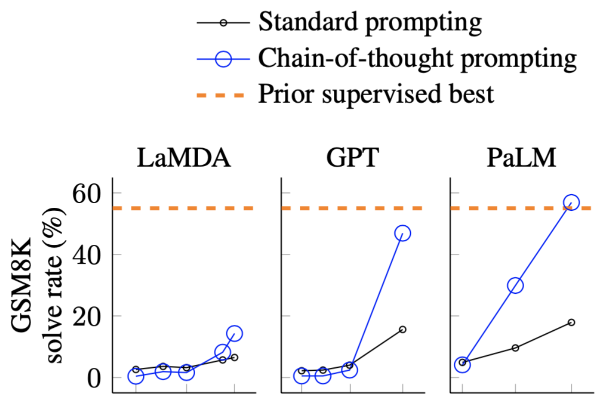


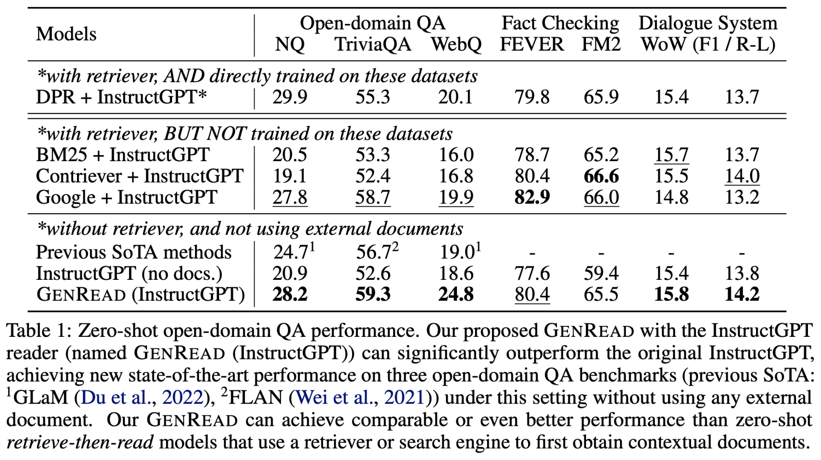
Figure 1: CoT Prompting Elicits Reasoning in LLMs

2. Emergent abilities contain complex reasoning. From Listing 1, it is nontrivial for a language model to solve problems in GSM8K dataset, due to the mixture of math and language. However, when the model size comes to over 100B, CoT outperforms standard answer-only prompting, and it is significantly better than previous fine-tuning methods.

```
Question:
Claire makes a 3 egg omelet every morning for breakfast. How many dozens
    of eggs will she eat in 4 weeks?
Claire makes a 3 egg omelet every morning.
There are 7 days in a week.
So in one week she will eat 3 * 7 = 21 eggs.
In 4 weeks she will eat 4 * 21 = 84 eggs.
There are 12 eggs in a dozen.
So 84 / 12 = 7.
The answer is 7.
```

Listing 1: One example in GSM8K dataset

3. Emergent abilities contain reasoning with knowledge. From Figure 2, we find that previous SOTA models require retrieving from outside knowledge sources. However, GPT-3 outperforms previous models without needing to retrieve, because it contains the knowledge in itself.

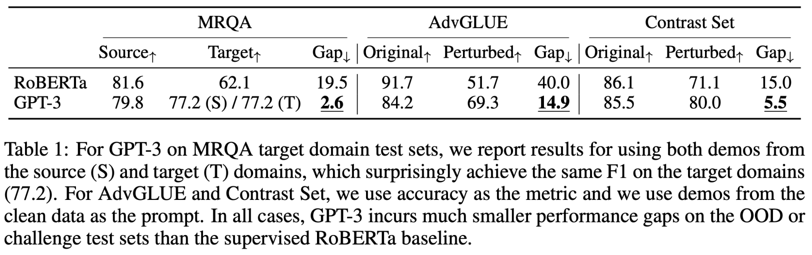


Models	Open-domain QA			Fact Checking		Dialogue System	
	NQ	TriviaQA	WebQ	FEVER	FM2	WoW (F1 / R-L)	
**with retriever, AND directly trained on these datasets*							
DPR + InstructGPT*	29.9	55.3	20.1	79.8	65.9	15.4	13.7
**with retriever, BUT NOT trained on these datasets*							
BM25 + InstructGPT	20.5	53.3	16.0	78.7	65.2	15.7	13.7
Contriever + InstructGPT	19.1	52.4	16.8	80.4	**66.6**	15.5	14.0
Google + InstructGPT	27.8	58.7	19.9	**82.9**	66.0	14.8	13.2
**without retriever, and not using external documents*							
Previous SoTA methods	24.7^1	56.7^2	19.0^1	-	-	-	-
InstructGPT (no docs.)	20.9	52.6	18.6	77.6	59.4	15.4	13.8
GENREAD (InstructGPT)	**28.2**	**59.3**	**24.8**	80.4	65.5	**15.8**	**14.2**

Table 1: Zero-shot open-domain QA performance. Our proposed GENREAD with the InstructGPT reader (named GENREAD (InstructGPT)) can significantly outperform the original InstructGPT, achieving new state-of-the-art performance on three open-domain QA benchmarks (previous SoTA: ^{1}GLaM (Du et al., 2022), ^{2}FLAN (Wei et al., 2021)) under this setting without using any external document. Our GENREAD can achieve comparable or even better performance than zero-shot *retrieve-then-read* models that use a retriever or search engine to first obtain contextual documents.

Figure 2: Complex reasoning

4. Emergent abilities contain out-of-distribution robustness. From 3, we find that although GPT-3 cannot outperform RoBERTa in the in-distribution setting, it outperforms RoBERTa in the out-of-distribution setting with a significantly smaller performance drop, so GPT-3 is more robust. □



	MRQA			AdvGLUE			Contrast Set		
	Source$_\uparrow$	Target$_\uparrow$	Gap$_\downarrow$	Original$_\uparrow$	Perturbed$_\uparrow$	Gap$_\downarrow$	Original$_\uparrow$	Perturbed$_\uparrow$	Gap$_\downarrow$
RoBERTa	81.6	62.1	19.5	91.7	51.7	40.0	86.1	71.1	15.0
GPT-3	79.8	77.2 (S) / 77.2 (T)	**2.6**	84.2	69.3	**14.9**	85.5	80.0	**5.5**

Table 1: For GPT-3 on MRQA target domain test sets, we report results for using both demos from the source (S) and target (T) domains, which surprisingly achieve the same F1 on the target domains (77.2). For AdvGLUE and Contrast Set, we use accuracy as the metric and we use demos from the clean data as the prompt. In all cases, GPT-3 incurs much smaller performance gaps on the OOD or challenge test sets than the supervised RoBERTa baseline.

Figure 3: Out-of-distribution robustness

1.3

(2 pts) Evaluate Qwen2.5-0.5B on **GSM8K** and **MATH** benchmark using different prompting methods: **zero-shot CoT**, and **few-shot** (with shot number $k = 5$). Report the accuracy of each setting.

Solution. See `eval_gsm8k.py` and `eval_math.py`. For accuracy, see Table 1.

We suppose the test data has been downloaded under the same location with codes.

How to run `eval_gsm8k.py`:

```
python3 eval_gsm8k.py --model Qwen/Qwen2.5-0.5B --data_file data/test/
    GSM8K_test.jsonl --mode both --tensor_parallel_size 1--batch_size 1 --
    k_shot 5 --max_tokens 256
```

How to run `eval_math.py`:

```
python3 eval_math.py --model Qwen/Qwen2.5-0.5B --data_file data/test/
    MATH_test.jsonl --mode zero (few) --tensor_parallel_size 1 --batch_size
     1
```

benchmark	**zero-shot CoT**	**few-shot ($k = 5$)**
GSM8K	27.27%	29.19%
MATH	22.74%	28.58%

Table 1: Accuracy

□

2 Q2

Explain and answer the following questions in your own words. You may refer to lecture notes for more details.

2.1

(2 pts) Briefly describe the three main filtering strategies in synthetic data pipelines: **Diversity filtering**, **Quality filtering**, and **Correctness filtering**. Explain one representative method or example for each (e.g., ROUGE-L, reward model, final answer verification).

Solution.

Diversity filtering: Remove the samples that are overly repetitive among each other or have highly similar meanings, to ensure a wider and more comprehensive data distribution and maximize the variety.

ROUGE-L: ROUGE-L measures the overlap between the candidate and the reference. *L* means longest common subsequence (LCS), which can evaluate similarity of sentence-level structure. A higher ROUGE-L score indicates greater meaning similarity.

Quality filtering: Eliminate the samples that are empty of content, full of nonsense or have confusing logic, and retain the high-quality expressions.

Reward model: It is trained to assign a scalar score to a model's output based on how well it satisfies human expectations. It is typically trained on a dataset of human preference rankings, where human evaluators rank several different model responses for the same prompt. RM learns to predict the ranking, effectively encoding human preference.

Correctness filtering: Ensure that the content generated by the model is correct and the answers are precise, and avoid spreading misinformation.

Final answer verification: It focuses on the final result of the model's generated response rather than the intermediate steps. If an LLM generates a solution to a math problem, the final numerical answer is extracted into an external, trusted calculator to confirm its accuracy.

If the LLM generates a piece of code, that code is executed in a sandbox environment, and the resulting output is checked against the correct output.

□

2.2

(3 pts) Write code to illustrate a simple **Reward-based Filtering** process:

1. Prepare 10 seed instruction–response pairs (can be manually written).
2. For each instruction, use an open/proprietary LLM (e.g., free models in OpenRouter) to generate candidate responses.
3. Assign a heuristic reward score to each response (e.g., weighted sum of response length, keyword match, and BLEU score, etc.).
4. Keep the top-1 response as the "filtered" sample.
5. Print all candidates with their scores and indicate the selected one.

Example output:

```
Prompt: Explain the difference between supervised and unsupervised
    learning.
Candidate 1: {xxxx}. Score = 0.85
Candidate 2: {xxxx}. Score = 0.78
Candidate 3: {xxxx}. Score = 0.74
...
Selected: Candidate 1
```

Solution. See `filtering.py`.

The output (only take prompt 1 as an example):

```
Prompt 1: Explain the difference between supervised and unsupervised
    learning.

Candidate 1: This question refers to fundamental ML concepts. Basically,
    ...
Score = 0.1575

Candidate 2: A concise explanation is as follows: ...
Score = 0.1000

Candidate 3: This question refers to fundamental ML concepts. Basically,
    ...
Score = 0.1575

Selected: Candidate 1
```

□