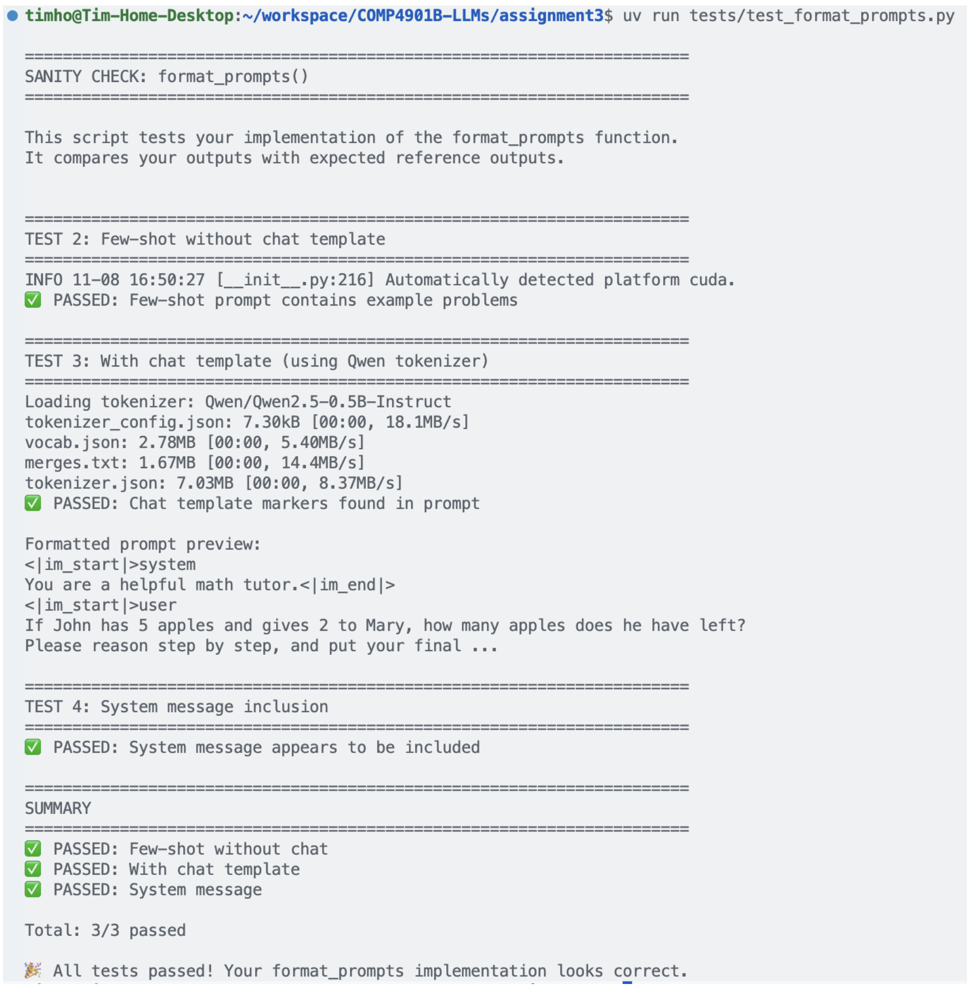
COMP4901B: Large Language Models

Assignment 3 Report

HE, Wenqian
Student ID: 20860896

November 15, 2025

1 Part 1 — VLLM Inference Implementation



```
● timho@Tim-Home-Desktop:~/workspace/COMP4901B-LLMs/assignment3$ uv run tests/test_format_prompts.py

======================================================================
SANITY CHECK: format_prompts()
======================================================================

This script tests your implementation of the format_prompts function.
It compares your outputs with expected reference outputs.


======================================================================
TEST 2: Few-shot without chat template
======================================================================
INFO 11-08 16:50:27 [__init__.py:216] Automatically detected platform cuda.
✅ PASSED: Few-shot prompt contains example problems

======================================================================
TEST 3: With chat template (using Qwen tokenizer)
======================================================================
Loading tokenizer: Qwen/Qwen2.5-0.5B-Instruct
tokenizer_config.json: 7.30kB [00:00, 18.1MB/s]
vocab.json: 2.78MB [00:00, 5.40MB/s]
merges.txt: 1.67MB [00:00, 14.4MB/s]
tokenizer.json: 7.03MB [00:00, 8.37MB/s]
✅ PASSED: Chat template markers found in prompt

Formatted prompt preview:
<|im_start|>system
You are a helpful math tutor.<|im_end|>
<|im_start|>user
If John has 5 apples and gives 2 to Mary, how many apples does he have left?
Please reason step by step, and put your final ...

======================================================================
TEST 4: System message inclusion
======================================================================
✅ PASSED: System message appears to be included

======================================================================
SUMMARY
======================================================================
✅ PASSED: Few-shot without chat
✅ PASSED: With chat template
✅ PASSED: System message

Total: 3/3 passed

🎉 All tests passed! Your format_prompts implementation looks correct.
```

Figure 1: VLLM Inference

1.1 Implementation

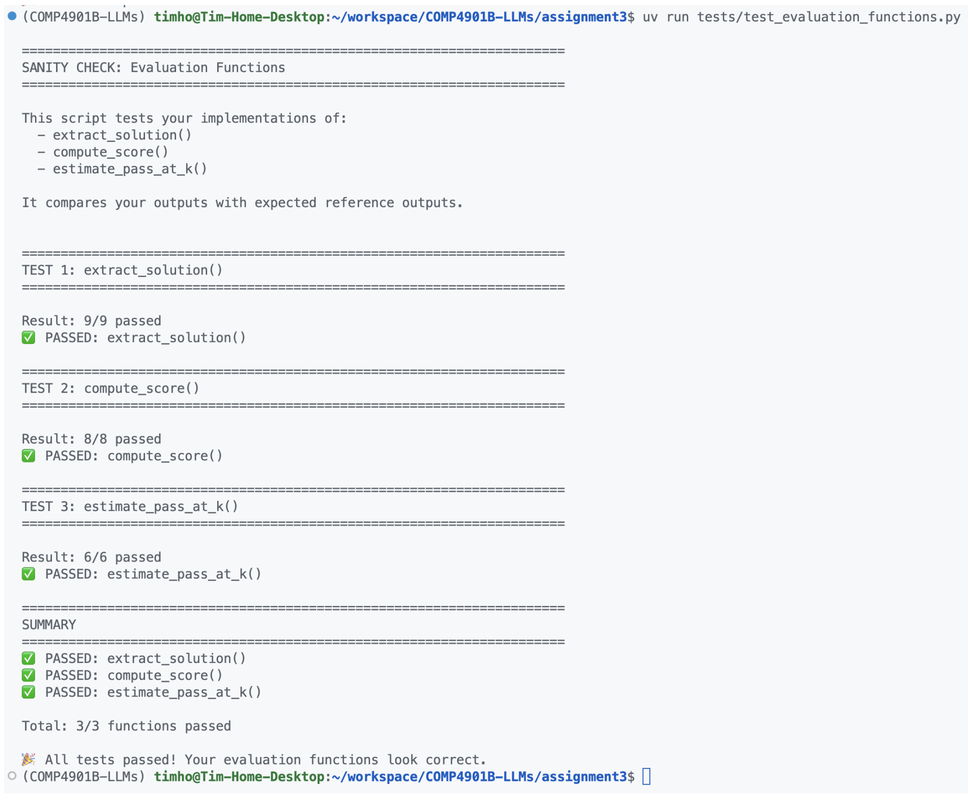
`format_prompts()` I followed the all instructions to implement this function. The quesiton is formatted using specified prompt template by default, then the formatted prompt is applied chat template from tokenizer with optional system message unless exception occurs, otherwise the raw formatted prompt without chat template is used.

`run_inference()` I followed the all instructions to implement this function. I pass all necessary parameters to LLM engine and sampling parameters. Specially, If the number of rollouts is greater than 1, but temperature is 0.0, the temperature is increased to 0.6 by default to avoid deterministic output. If the number of rollouts is 1, the temperature is set to 0.0 to avoid stochastic output. Otherwise the temperature is set to the specified temperature.

1.2 Answer the question

We need temperature > 0 when generating multiple rollouts because when temperature is 0, the model will generate the same output for each rollout, then there is no meaning to generate multiple rollouts. So we should set temperature > 0 to generate diverse outputs.

2 Part 2 — Answer Verification Implementation



```
(COMP4901B-LLMs) timho@Tim-Home-Desktop:~/workspace/COMP4901B-LLMs/assignment3$ uv run tests/test_evaluation_functions.py

======================================================================
SANITY CHECK: Evaluation Functions
======================================================================

This script tests your implementations of:
  - extract_solution()
  - compute_score()
  - estimate_pass_at_k()

It compares your outputs with expected reference outputs.


======================================================================
TEST 1: extract_solution()
======================================================================

Result: 9/9 passed
✅ PASSED: extract_solution()

======================================================================
TEST 2: compute_score()
======================================================================

Result: 8/8 passed
✅ PASSED: compute_score()

======================================================================
TEST 3: estimate_pass_at_k()
======================================================================

Result: 6/6 passed
✅ PASSED: estimate_pass_at_k()

======================================================================
SUMMARY
======================================================================
✅ PASSED: extract_solution()
✅ PASSED: compute_score()
✅ PASSED: estimate_pass_at_k()

Total: 3/3 functions passed

🎉 All tests passed! Your evaluation functions look correct.
(COMP4901B-LLMs) timho@Tim-Home-Desktop:~/workspace/COMP4901B-LLMs/assignment3$
```

Figure 2: Answer Verification

2.1 Implementation

```
BOX_PATTERN = r'\\boxed\{([^}]+)\}'
NUMBER_PATTERN = r'(-?(?:\d{1,3}(?:,\d{3})*|\d+)(?:\.\d+)?)'
```

`extract_solution()`: I first try to extract the answer in boxed format using the pattern `BOX_PATTERN`, then if not found, I try to extract the answer in number format using the pattern `NUMBER_PATTERN`. The last extracted answer is returned without commas and dollar signs. If both failed, I return None.

`compute_score()`: I casted the extracted answer and ground truth to float and back to string to compare the values. This operation is performed in try-except block to handle non-numeric values. If the comparison is successful, it returned the comparison result otherwise 0 is returned.

2.2 Derivation of the pass@k metric

$$\begin{aligned}
\text{pass@k} &= P(\text{at least one correct}) \\
&= 1 - P(\text{all are wrong}) \\
&= 1 - \frac{\binom{n-c}{k}}{\binom{n}{k}} \\
&= 1 - \frac{(n-c)!}{k!(n-c-k)!} \times \frac{k!(n-k)!}{n!} \\
&= 1 - \frac{(n-c-k+1)(n-c-k+2)\cdots(n-c)}{(n-k+1)(n-k+2)\cdots(n)} \\
&= 1 - \prod_{i=1}^{k} \frac{n-c-k+i}{n-k+i}
\end{aligned}$$

2.3 Answer the question

```
ratios = np.arange(n-c-k+1, n-c+1) / np.arange(n-k+1, n+1)
pass_at_k_values[i] = 1 - np.prod(ratios)
```

The numerical stability is achieved by using the product formulation of the pass@k metric. I created a 1d array of

$$n-c-k+1, n-c-k+2, \cdots, n-c$$

and a 1d array of

$$n-k+1, n-k+2, \cdots, n$$

then divide the corresponding elements to get the 1d array of ratios. Then the pass@k is 1 minus the product of the ratios. This avoids the computation of large factorials and division of large numbers.

3 Part 3 — LoRA Training Implementation

```python
def _resolve_lora_target_modules(self) -> List[str]:
    VALID_TARGETS_BY_MODEL = {
        "qwen3": {"q_proj", "k_proj", "v_proj", "o_proj", "gate_proj",
           "up_proj", "down_proj"},
        "llama": {'o_proj', 'k_proj', 'up_proj', 'gate_proj', 'v_proj',
           'q_proj', 'down_proj'},
        "mistral": {'k_proj', 'o_proj', 'down_proj', 'q_proj', 'up_proj
           ', 'gate_proj', 'v_proj'},
        "opt": {'k_proj', 'q_proj', 'fc1', 'v_proj', 'out_proj', 'fc2'}
    }

    model_type: str = self.model.config.model_type

    if self.lora_args.lora_target_modules:
        targets = set(self.lora_args.lora_target_modules)
    elif model_type in VALID_TARGETS_BY_MODEL:
        targets = VALID_TARGETS_BY_MODEL[model_type]
    else:
        raise ValueError(f"No valid target modules found for model type
           : {model_type}")

    available_modules = {name.split(".")[-1] for name, module in self.
       model.named_modules() if isinstance(module, nn.Linear)}

    assert targets <= available_modules, f"Invalid target modules: {
       targets} not in {available_modules}"
    valid_targets = list(targets)

    return valid_targets
```

3.1 Detect architecture and select target modules

I use `self.model.config.model_type` to detect the architecture of the model. I downloaded `Qwen/Qwen3-0.6B`, `unsloth/Llama-3.2-1B`, `unsloth/mistral-7b-bnb-4bit` and `facebook/opt-125m` models from Hugging Face, then printed the model type string, model architecture, and linear module names. Then I picked the name of linear modules in decoder layers for each model. They are the default valid target modules for forementioned models if no target modules are specified. Details can be found in the `scripts/model.ipynb` file.

3.2 Answer the question

In decoder-only transformer models, the attention layers are used to correlate different parts of the input sequence, i.e. attend to appropriate positions. The FFN layers are used to store the knowledge, such as math operation knowledge. Finetuning lora on attention and FFN layers can help the model focus on the important numbers and quesiton intention keywords for GSM8K-like math problems, then match the learned patterns for each token in FFN layers to perform math reasoning.

If we only apply lora to attention layers, although it might improve the model's attention on some important number of keywords, but it cannot learn new patterns, such as for which kind of question should it apply division or multiplication.

Moreover, since the model cannot learn new pattern, it might not be able to extract better features in lower layers through FFN, then higher attention layers cannot pay attention to good positions since they are not similar in terms of features, even though they might relevant in eyes of humans.

4 Part 4 — Part 4 — Self-Training Execution

4.1 Training Configuration Summary

I trained the models for 4 iterations using the following configuration, which is also the default configuration.

```
NUM_ITERATIONS=4
MAX_TOKENS=512
N_ROLLOUTS=8
ENABLE_THINKING=false
N_QUERIES=2000

LEARNING_RATE=2e-5
TOTAL_BATCH_SIZE=128
NUM_EPOCHS=1
SAVE_STEPS=30
LORA_R=64
```

4.2 Metrics

See table 1 for the metrics across training iterations and figure 3 for the accuracy improvement across iterations. The number of coorect solutions found and accuracy is collected by `run_gsm8k_eval.sh` script with single rollout. The pass@k metrics are collected by `run_gsm8k_eval.sh` script with multiple rollouts. The training loss is collected in the end of each iteration. Detailed log can be found at `ckpt/` and `results/` directories.

Iteration	#Correct	Accuracy	Pass@1	Pass@4	Pass@8	Training Loss
0 (baseline)	803	60.88%	0.6070	0.7810	0.8500	—
1	838	63.53%	0.6300	0.8020	0.8480	0.2865
2	841	63.76%	0.6080	0.8130	**0.8620**	0.2874
3	836	63.38%	**0.6320**	**0.8030**	0.8580	0.2875
4	**854**	**64.75%**	0.6310	0.8040	0.8570	**0.2862**

Table 1: Performance metrics across training iterations. Bold values indicate the best performance in each column.

4.3 Analysis

Self-training improves the accuracy from 60.88% to 64.75% across 4 iterations. By training on the correct solutions, the model learns what's the potential correct reasoning process that leads to correct answers from the output sampled from the model itself, and tend to generate reasoning process similar to them in the future, therefore solving similar problems more accurately.

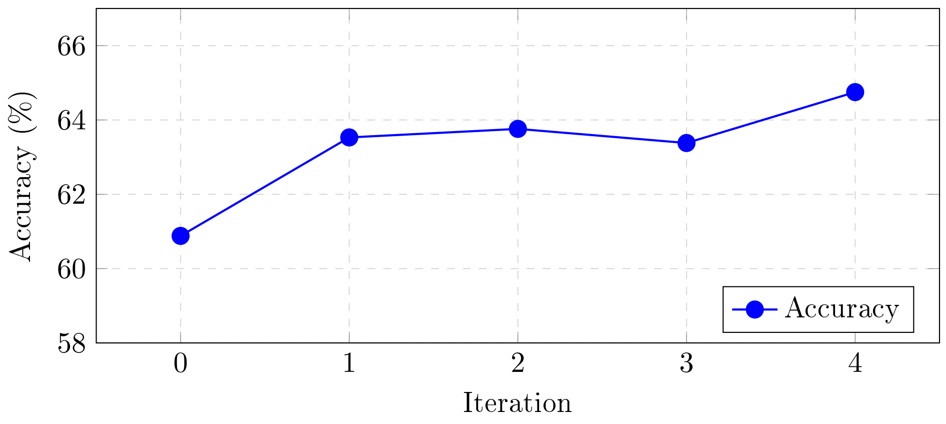


Figure 3: Accuracy improvement across iterations

5 Part 5 — Final Evaluation and Analysis

5.1 Baseline Evaluation

The baseline accuracy is 60.88% as shown in the figure 4a. The detailed evaluation results can be found at `results/baseline/` directory.

5.2 Trained Model Evaluation

Please see table 2 for the evaluation results across different hyperparameter variations and training iterations. The detailed evaluation results can be found at `results/` directory. All screenshots can be found in the appendix.

Ablation Study: Based on the evaluation results, within one iteration, the best performance is achieved when lowering the batch size to 64, attaining 63.84% accuracy. The other settings, including decreasing learning rate, increasing learning rate, decreasing queries, increasing queries, increasing batch size, are all worse than the default settings.

Best Performance: The best performance is achieved by running more iterations on the default settings, which is 64.75% accuracy.

5.3 Analysis

Please see table 3 for the detailed analysis of the avg@k and pass@k metrics across hyperparameter variations and training iterations. The detailed evaluation results can be found at `results/` directory. All screenshots can be found in the appendix.

Trend: For avg@k metrics, we can see that for the first iteration, the avg@k tend to be lower as k increase, but in higher iteration, the avg@k tends to be stable or slightly increase. However,

Model	Iteration	BSZ	LR	N_QUERIES	Accuracy
default	1	128	2e-5	2000	63.53%
↓ lr	1	128	1e-5	2000	61.56%
↑ lr	1	128	4e-5	2000	62.55%
↓ queries	1	128	2e-5	1500	63.31%
↑ queries	1	128	2e-5	3000	62.17%
↓ batch size	1	64	2e-5	2000	**63.84%**
↑ batch size	1	256	2e-5	2000	62.70%
default iter 1	1	128	2e-5	2000	63.53%
default iter 2	2	128	2e-5	2000	63.76%
default iter 3	3	128	2e-5	2000	63.38%
default iter 4	4	128	2e-5	2000	**64.75%**

Table 2: Hyperparameter ablation study and iterative training results. Bold values indicate best performance within each section.

for pass@k metrics, we can see that for all iterations, the pass@k tends to be higher as k increase, but higher iteration does not indicate better pass@k metrics.

Gap between avg@k and pass@k metrics: In baseline model, and also all the trained models, there is a gap between avg@k and pass@k metrics, indicating although the accuracy may not be high enough, we by trying multiple times, the model is still able to provide correct answer with high probability, even for some cases, the correct answer is not the first one. Therefore, pass@k metrics with high k is the potential value of the accuracy to achieve by the model after self-training.

Comparison before and after training: All metrics after training are better than the baseline (for default settings), indicating that self-training is effective. By training on the correct solutions, the model learns what's the potential correct reasoning process that leads to correct answers from the output sampled from the model itself, and tend to generate reasoning process similar to them in the future, therefore solving similar problems more accurately.

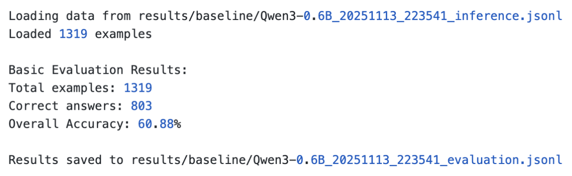
Diversity and Reliability: When training with default settings within one iteration, the improvement of pass@k metrics is lower and lower as k increase, although avg@k metrics improves with roughly fixed amount. This indicates that the model is more reliable in general to provide correct answer without extensive retry, but the output is less diverse, less likely to get correct answer surprisingly after multiple attempts. Since pass@k metrics improved less significantly compared to avg@k metrics, the model is less diverse after training.

Model	**Iter**	**avg@1**	**avg@4**	**avg@8**	**pass@1**	**pass@4**	**pass@8**
baseline	0	0.6070	0.5962	0.5966	0.6070	0.7810	0.8500
default	1	**0.6300**	0.6148	**0.6146**	**0.6300**	0.8020	0.8480
↓ lr	1	0.6230	0.6110	0.6095	0.6230	0.7940	0.8480
↑ lr	1	0.6030	0.6067	0.6089	0.6030	0.7830	0.8330
↓ queries	1	0.6090	0.6072	0.6125	0.6090	0.7880	0.8440
↑ queries	1	0.6110	0.6140	0.6121	0.6110	**0.8030**	0.8480
↓ batch size	1	0.6260	**0.6152**	0.6134	0.6260	0.8020	**0.8500**
↑ batch size	1	0.6170	0.6098	0.6076	0.6170	0.7880	0.8390
default iter 1	1	0.6300	0.6148	0.6146	0.6300	0.8020	0.8480
default iter 2	2	0.6080	0.6302	0.6300	0.6080	**0.8130**	**0.8620**
default iter 3	3	**0.6320**	**0.6360**	0.6309	**0.6320**	0.8030	0.8580
default iter 4	4	0.6310	0.6330	**0.6350**	0.6310	0.8040	0.8570

Table 3: Detailed analysis of average and pass@k metrics across hyperparameter variations and training iterations. Bold values indicate best performance within each section.

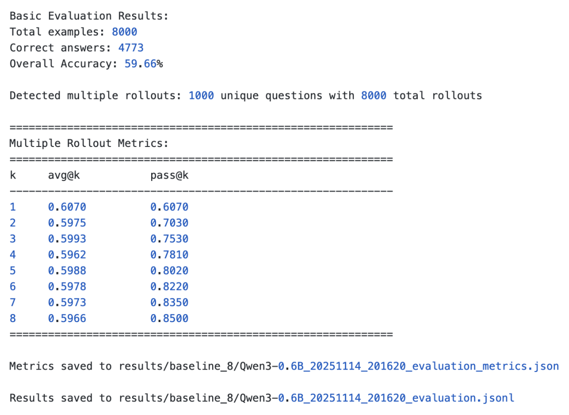
6 Appendix

6.1 Ablation Study Evaluation Results



```
Loading data from results/baseline/Qwen3-0.6B_20251113_223541_inference.jsonl
Loaded 1319 examples

Basic Evaluation Results:
Total examples: 1319
Correct answers: 803
Overall Accuracy: 60.88%

Results saved to results/baseline/Qwen3-0.6B_20251113_223541_evaluation.jsonl
```

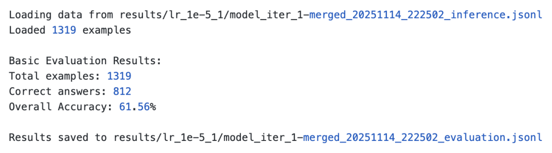
(a) 1 rollout



```
Basic Evaluation Results:
Total examples: 8000
Correct answers: 4773
Overall Accuracy: 59.66%

Detected multiple rollouts: 1000 unique questions with 8000 total rollouts

============================================================
Multiple Rollout Metrics:
============================================================
k     avg@k           pass@k
------------------------------------------------------------
1     0.6070          0.6070
2     0.5975          0.7030
3     0.5993          0.7530
4     0.5962          0.7810
5     0.5988          0.8020
6     0.5978          0.8220
7     0.5973          0.8350
8     0.5966          0.8500
============================================================

Metrics saved to results/baseline_8/Qwen3-0.6B_20251114_201620_evaluation_metrics.json

Results saved to results/baseline_8/Qwen3-0.6B_20251114_201620_evaluation.jsonl
```

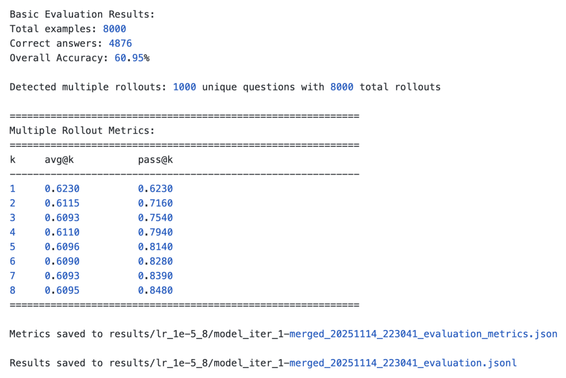
(b) 8 rollouts

Figure 4: Baseline evaluation results with different numbers of rollouts



```
Loading data from results/lr_1e-5_1/model_iter_1-merged_20251114_222502_inference.jsonl
Loaded 1319 examples

Basic Evaluation Results:
Total examples: 1319
Correct answers: 812
Overall Accuracy: 61.56%

Results saved to results/lr_1e-5_1/model_iter_1-merged_20251114_222502_evaluation.jsonl
```

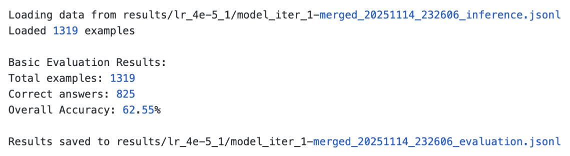
(a) 1 rollout



```
Basic Evaluation Results:
Total examples: 8000
Correct answers: 4876
Overall Accuracy: 60.95%

Detected multiple rollouts: 1000 unique questions with 8000 total rollouts

============================================================
Multiple Rollout Metrics:
============================================================
k     avg@k           pass@k
------------------------------------------------------------
1     0.6230          0.6230
2     0.6115          0.7160
3     0.6093          0.7540
4     0.6110          0.7940
5     0.6096          0.8140
6     0.6090          0.8280
7     0.6093          0.8390
8     0.6095          0.8480
============================================================

Metrics saved to results/lr_1e-5_8/model_iter_1-merged_20251114_223041_evaluation_metrics.json

Results saved to results/lr_1e-5_8/model_iter_1-merged_20251114_223041_evaluation.jsonl
```

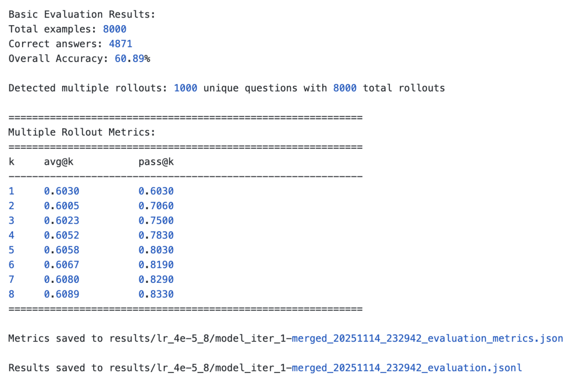
(b) 8 rollouts

Figure 5: Decreasing learning rate to 1e-5 evaluation results with different numbers of rollouts



```
Loading data from results/lr_4e-5_1/model_iter_1-merged_20251114_232606_inference.jsonl
Loaded 1319 examples

Basic Evaluation Results:
Total examples: 1319
Correct answers: 825
Overall Accuracy: 62.55%

Results saved to results/lr_4e-5_1/model_iter_1-merged_20251114_232606_evaluation.jsonl
```

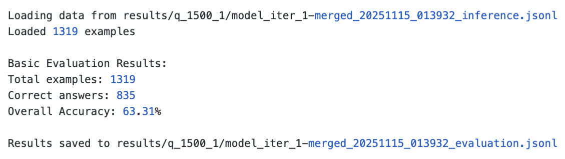
(a) 1 rollout



```
Basic Evaluation Results:
Total examples: 8000
Correct answers: 4871
Overall Accuracy: 60.89%

Detected multiple rollouts: 1000 unique questions with 8000 total rollouts

============================================================
Multiple Rollout Metrics:
============================================================
k     avg@k           pass@k
------------------------------------------------------------
1     0.6030          0.6030
2     0.6005          0.7060
3     0.6023          0.7500
4     0.6052          0.7830
5     0.6058          0.8030
6     0.6067          0.8190
7     0.6080          0.8290
8     0.6089          0.8330
============================================================

Metrics saved to results/lr_4e-5_8/model_iter_1-merged_20251114_232942_evaluation_metrics.json

Results saved to results/lr_4e-5_8/model_iter_1-merged_20251114_232942_evaluation.jsonl
```

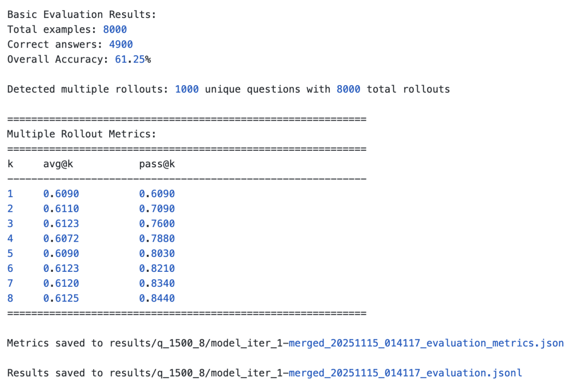
(b) 8 rollouts

Figure 6: Increasing learning rate to 4e-5 evaluation results with different numbers of rollouts



```
Loading data from results/q_1500_1/model_iter_1-merged_20251115_013932_inference.jsonl
Loaded 1319 examples

Basic Evaluation Results:
Total examples: 1319
Correct answers: 835
Overall Accuracy: 63.31%

Results saved to results/q_1500_1/model_iter_1-merged_20251115_013932_evaluation.jsonl
```

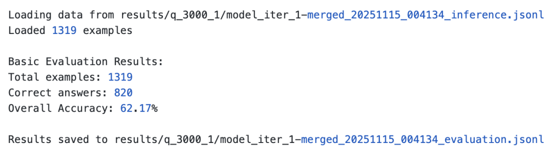
(a) 1 rollout



```
Basic Evaluation Results:
Total examples: 8000
Correct answers: 4900
Overall Accuracy: 61.25%

Detected multiple rollouts: 1000 unique questions with 8000 total rollouts

============================================================
Multiple Rollout Metrics:
============================================================
k     avg@k          pass@k
------------------------------------------------------------
1     0.6090         0.6090
2     0.6110         0.7090
3     0.6123         0.7600
4     0.6072         0.7880
5     0.6090         0.8030
6     0.6123         0.8210
7     0.6120         0.8340
8     0.6125         0.8440
============================================================

Metrics saved to results/q_1500_8/model_iter_1-merged_20251115_014117_evaluation_metrics.json

Results saved to results/q_1500_8/model_iter_1-merged_20251115_014117_evaluation.jsonl
```

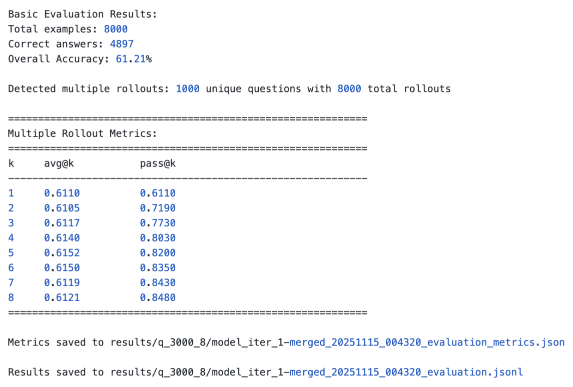
(b) 8 rollouts

Figure 7: Decreasing queries to 1500 evaluation results with different numbers of rollouts



```
Loading data from results/q_3000_1/model_iter_1-merged_20251115_004134_inference.jsonl
Loaded 1319 examples

Basic Evaluation Results:
Total examples: 1319
Correct answers: 820
Overall Accuracy: 62.17%

Results saved to results/q_3000_1/model_iter_1-merged_20251115_004134_evaluation.jsonl
```

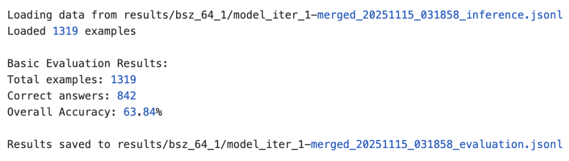
(a) 1 rollout



```
Basic Evaluation Results:
Total examples: 8000
Correct answers: 4897
Overall Accuracy: 61.21%

Detected multiple rollouts: 1000 unique questions with 8000 total rollouts

============================================================
Multiple Rollout Metrics:
============================================================
k     avg@k          pass@k
------------------------------------------------------------
1     0.6110         0.6110
2     0.6105         0.7190
3     0.6117         0.7730
4     0.6140         0.8030
5     0.6152         0.8200
6     0.6150         0.8350
7     0.6119         0.8430
8     0.6121         0.8480
============================================================

Metrics saved to results/q_3000_8/model_iter_1-merged_20251115_004320_evaluation_metrics.json

Results saved to results/q_3000_8/model_iter_1-merged_20251115_004320_evaluation.jsonl
```

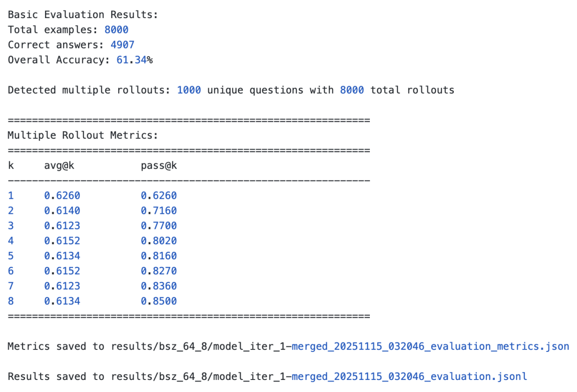
(b) 8 rollouts

Figure 8: Increasing queries to 3000 evaluation results with different numbers of rollouts



```
Loading data from results/bsz_64_1/model_iter_1-merged_20251115_031858_inference.jsonl
Loaded 1319 examples

Basic Evaluation Results:
Total examples: 1319
Correct answers: 842
Overall Accuracy: 63.84%

Results saved to results/bsz_64_1/model_iter_1-merged_20251115_031858_evaluation.jsonl
```

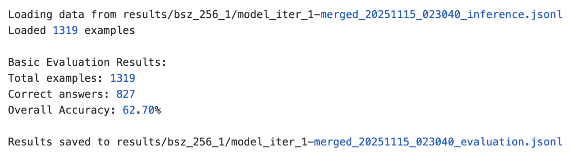
(a) 1 rollout



```
Basic Evaluation Results:
Total examples: 8000
Correct answers: 4907
Overall Accuracy: 61.34%

Detected multiple rollouts: 1000 unique questions with 8000 total rollouts

============================================================
Multiple Rollout Metrics:
============================================================
k     avg@k          pass@k
------------------------------------------------------------
1     0.6260         0.6260
2     0.6140         0.7160
3     0.6123         0.7700
4     0.6152         0.8020
5     0.6134         0.8160
6     0.6152         0.8270
7     0.6123         0.8360
8     0.6134         0.8500
============================================================

Metrics saved to results/bsz_64_8/model_iter_1-merged_20251115_032046_evaluation_metrics.json

Results saved to results/bsz_64_8/model_iter_1-merged_20251115_032046_evaluation.jsonl
```

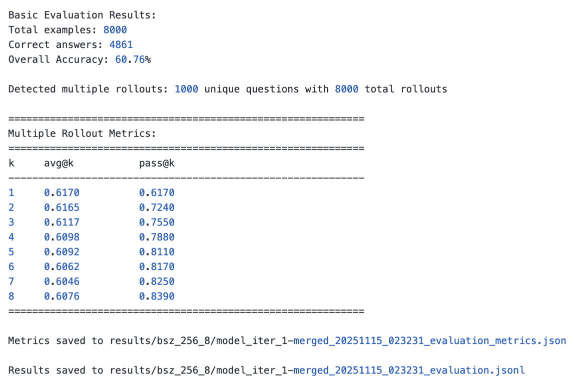
(b) 8 rollouts

Figure 9: Decreasing batch size to 64 evaluation results with different numbers of rollouts



```
Loading data from results/bsz_256_1/model_iter_1-merged_20251115_023040_inference.jsonl
Loaded 1319 examples

Basic Evaluation Results:
Total examples: 1319
Correct answers: 827
Overall Accuracy: 62.70%

Results saved to results/bsz_256_1/model_iter_1-merged_20251115_023040_evaluation.jsonl
```

(a) 1 rollout

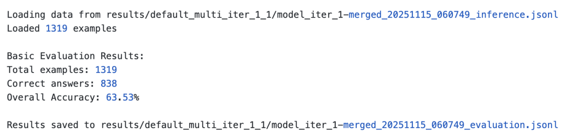


```
Basic Evaluation Results:
Total examples: 8000
Correct answers: 4861
Overall Accuracy: 60.76%

Detected multiple rollouts: 1000 unique questions with 8000 total rollouts

============================================================
Multiple Rollout Metrics:
============================================================
k     avg@k           pass@k
------------------------------------------------------------
1     0.6170          0.6170
2     0.6165          0.7240
3     0.6117          0.7550
4     0.6098          0.7880
5     0.6092          0.8110
6     0.6062          0.8170
7     0.6046          0.8250
8     0.6076          0.8390
============================================================

Metrics saved to results/bsz_256_8/model_iter_1-merged_20251115_023231_evaluation_metrics.json

Results saved to results/bsz_256_8/model_iter_1-merged_20251115_023231_evaluation.jsonl
```

(b) 8 rollouts

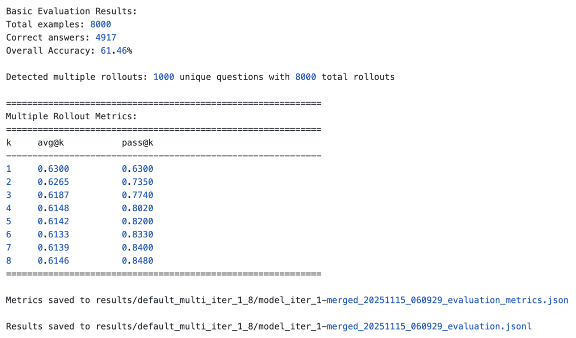
Figure 10: Increasing batch size to 256 evaluation results with different numbers of rollouts

6.2 Multi-Iteration Evaluation Results



```
Loading data from results/default_multi_iter_1_1/model_iter_1-merged_20251115_060749_inference.jsonl
Loaded 1319 examples

Basic Evaluation Results:
Total examples: 1319
Correct answers: 838
Overall Accuracy: 63.53%

Results saved to results/default_multi_iter_1_1/model_iter_1-merged_20251115_060749_evaluation.jsonl
```

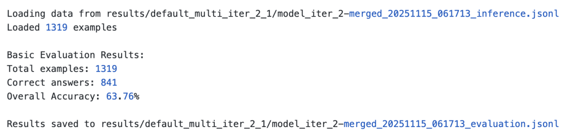
(a) 1 rollout



```
Basic Evaluation Results:
Total examples: 8000
Correct answers: 4917
Overall Accuracy: 61.46%

Detected multiple rollouts: 1000 unique questions with 8000 total rollouts

============================================================
Multiple Rollout Metrics:
============================================================
k     avg@k          pass@k
------------------------------------------------------------
1     0.6300         0.6300
2     0.6265         0.7350
3     0.6187         0.7740
4     0.6148         0.8020
5     0.6142         0.8200
6     0.6133         0.8330
7     0.6139         0.8400
8     0.6146         0.8480
============================================================

Metrics saved to results/default_multi_iter_1_8/model_iter_1-merged_20251115_060929_evaluation_metrics.json

Results saved to results/default_multi_iter_1_8/model_iter_1-merged_20251115_060929_evaluation.jsonl
```

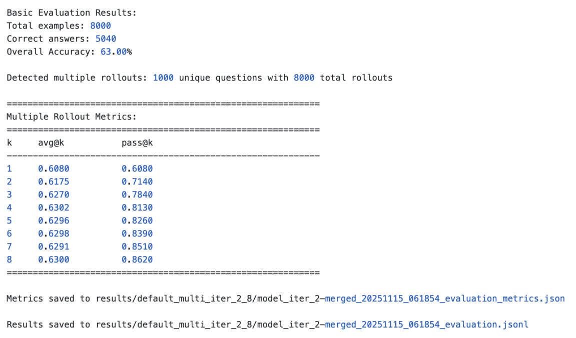
(b) 8 rollouts

Figure 11: Default 1 iteration evaluation results with different numbers of rollouts



```
Loading data from results/default_multi_iter_2_1/model_iter_2-merged_20251115_061713_inference.jsonl
Loaded 1319 examples

Basic Evaluation Results:
Total examples: 1319
Correct answers: 841
Overall Accuracy: 63.76%

Results saved to results/default_multi_iter_2_1/model_iter_2-merged_20251115_061713_evaluation.jsonl
```

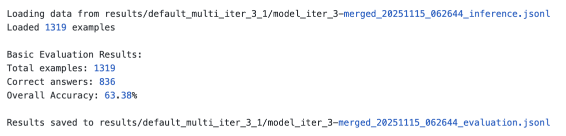
(a) 1 rollout



```
Basic Evaluation Results:
Total examples: 8000
Correct answers: 5040
Overall Accuracy: 63.00%

Detected multiple rollouts: 1000 unique questions with 8000 total rollouts

============================================================
Multiple Rollout Metrics:
============================================================
k     avg@k          pass@k
------------------------------------------------------------
1     0.6080         0.6080
2     0.6175         0.7140
3     0.6270         0.7840
4     0.6302         0.8130
5     0.6296         0.8260
6     0.6298         0.8390
7     0.6291         0.8510
8     0.6300         0.8620
============================================================

Metrics saved to results/default_multi_iter_2_8/model_iter_2-merged_20251115_061854_evaluation_metrics.json

Results saved to results/default_multi_iter_2_8/model_iter_2-merged_20251115_061854_evaluation.jsonl
```

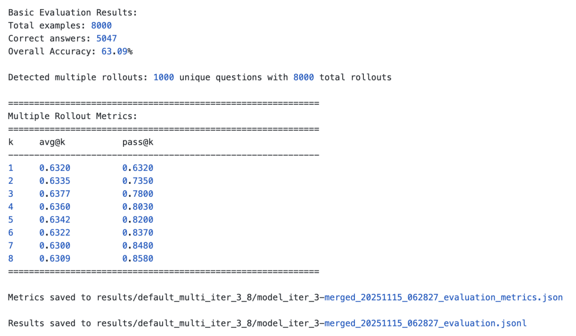
(b) 8 rollouts

Figure 12: Default 2 iterations evaluation results with different numbers of rollouts



```
Loading data from results/default_multi_iter_3_1/model_iter_3-merged_20251115_062644_inference.jsonl
Loaded 1319 examples

Basic Evaluation Results:
Total examples: 1319
Correct answers: 836
Overall Accuracy: 63.38%

Results saved to results/default_multi_iter_3_1/model_iter_3-merged_20251115_062644_evaluation.jsonl
```

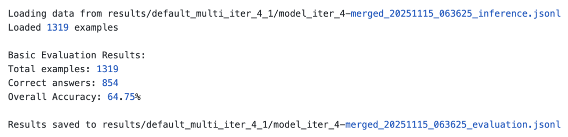
(a) 1 rollout



```
Basic Evaluation Results:
Total examples: 8000
Correct answers: 5047
Overall Accuracy: 63.09%

Detected multiple rollouts: 1000 unique questions with 8000 total rollouts

============================================================
Multiple Rollout Metrics:
============================================================
k     avg@k          pass@k
------------------------------------------------------------
1     0.6320         0.6320
2     0.6335         0.7350
3     0.6377         0.7800
4     0.6360         0.8030
5     0.6342         0.8200
6     0.6322         0.8370
7     0.6300         0.8480
8     0.6309         0.8580
============================================================

Metrics saved to results/default_multi_iter_3_8/model_iter_3-merged_20251115_062827_evaluation_metrics.json

Results saved to results/default_multi_iter_3_8/model_iter_3-merged_20251115_062827_evaluation.jsonl
```

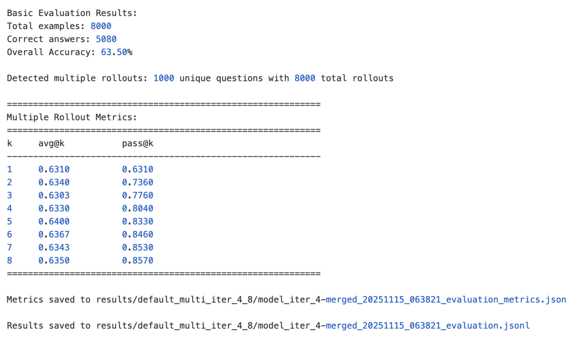
(b) 8 rollouts

Figure 13: Default 3 iterations evaluation results with different numbers of rollouts



```
Loading data from results/default_multi_iter_4_1/model_iter_4-merged_20251115_063625_inference.jsonl
Loaded 1319 examples

Basic Evaluation Results:
Total examples: 1319
Correct answers: 854
Overall Accuracy: 64.75%

Results saved to results/default_multi_iter_4_1/model_iter_4-merged_20251115_063625_evaluation.jsonl
```

(a) 1 rollout



```
Basic Evaluation Results:
Total examples: 8000
Correct answers: 5080
Overall Accuracy: 63.50%

Detected multiple rollouts: 1000 unique questions with 8000 total rollouts

============================================================
Multiple Rollout Metrics:
============================================================
k     avg@k           pass@k
------------------------------------------------------------
1     0.6310          0.6310
2     0.6340          0.7360
3     0.6303          0.7760
4     0.6330          0.8040
5     0.6400          0.8330
6     0.6367          0.8460
7     0.6343          0.8530
8     0.6350          0.8570
============================================================

Metrics saved to results/default_multi_iter_4_8/model_iter_4-merged_20251115_063821_evaluation_metrics.json

Results saved to results/default_multi_iter_4_8/model_iter_4-merged_20251115_063821_evaluation.jsonl
```

(b) 8 rollouts

Figure 14: Default 4 iterations evaluation results with different numbers of rollouts