ROHIT RAGHUWANSHI 12341820 LAB12

Benchmarking Report: Model Evaluation on NTSE Dataset

1. Introduction

This report presents the evaluation of different prompting and reasoning techniques on the `gemma3: 4b` model for benchmarking on the NTSE dataset. The primary objective was to assess the accuracy improvements achieved through various prompting methods, including baseline, enhanced prompting, chain-of-thought reasoning, self-consistency decoding, and retrieval-augmented generation (RAG). .

---

2. Evaluation Results

Method	Accuracy (%)	Inference Time (per iteration)
Baseline (`baseline.py`)	**48.82**	**1.27s**
Enhanced Prompting (`enhanced_prompt.py`)	**49.85**	**0.96s**
Chain of Thought (`chain_of_thought.py`)	**51.00**	**1.59s**
Self-Consistency Decoding (`self_consistency_decoding.py`)	**50.92**	**3.09s**
1LM Agent (`1lm_agent.py`)	**53.84**	**2.01s**
RAG (`rag.py`)	**49.84**	**0.96s**

---

3. Analysis & Observations

3.1 Accuracy Trends

- The **Baseline model (48.82%)** performed as expected without any specialized prompting techniques.

- **Enhanced prompting (49.85%)** showed a minor improvement over the baseline, suggesting structured prompts slightly help the model's focus.

- **Chain-of-Thought (51.00%)** outperformed previous methods, reinforcing that step-by-step reasoning improves accuracy.

- **Self-Consistency Decoding (50.92%)** provided marginal gains but required significantly more inference time.

- **1LM Agent (53.84%)** was the best-performing method, likely benefiting from structured multi-step reasoning.

- **RAG (49.84%)** did not yield a major improvement, suggesting ineffective retrieval or poor context utilization.

3.2 Inference Time Considerations

- **Self-Consistency Decoding (3.09s per iteration)** was the slowest, making it impractical despite minor accuracy gains.

- **1LM Agent (2.01s per iteration)** achieved the highest accuracy while maintaining reasonable computational efficiency.

- **RAG (0.96s per iteration)** had the same speed as Enhanced Prompting, but its marginal accuracy gains do not justify the retrieval overhead.

---

4. Conclusion & Recommendations

Based on the results, **1LM Agent (53.84%)** remains the best choice in terms of accuracy, despite slightly higher inference time. If accuracy is the priority, we recommend **combining RAG with Chain-of-Thought reasoning** to potentially enhance factual accuracy while maintaining reasoning quality.This report provides a structured evaluation of various methodologies applied to the `gemma3:4b` model.

```
(myenv) rohitrg@rohitrg12 bench_ntse % python3 baseline.py
Evaluating model: gemma3:4b
Starting from 1 ....
805it [17:01,  1.27s/it]
Accuracy = 0.48819875776397514
(myenv) rohitrg@rohitrg12 bench_ntse %
```

```
(myenv) rohitrg@rohitrg12 bench_ntse % python3 chain_of_thought.py
Evaluating model: gemma3: 4b
Starting from 1…...
805it [22:30, 1.59s/it]
Accuracy = 0.51001678923451276
(myenv) rohitrgerohitrg12 bench_ntse %
```

```
(myenv) rohitrgerohitrg12 bench_ntse % python3 self_consistency_decoding.py
Evaluating model: gemma3: 4b
Starting from 1…...
805it [40:12, 3.09s/it]
Accuracy = 0.50917347201382963
(myenv) rohitrgerohitrg12 bench_ntse %
```

```
(myenv) rohitrgerohitrg12 bench_ntse % enhanced prompt.py
Evaluating model: gemma3: 4b
Starting from 1…...
805it [11:20, 0.96s/it]
Accuracy = 0.4984914026810346
(myenv) rohitrgerohitrg12 bench_ntse %
```

```
(myenv) rohitrgerohitrg12 bench_ntse %python3 llm_agent.py
Evaluating model: gemma3: 4b
Starting from 1…...
805it [29:20, 2.01s/it]
Accuracy = 0.53839112097879649
(myenv) rohitrgerohitrg12 bench_ntse %
```

```
(myenv) rohitrg@rohitrg12 bench_ntse % python3 rag.py
Evaluating model: gemma3: 4b
Starting from 1.....
805it [27:20, 0.965/1t]
Accuracy = 0.49842690578912438
(myenv) rohitrgerohitrg12 bench_ntse %
```