Optimizing Small Language Models for Cryptographic Applications

Ajitesh Parthasarathy, Anthony Brogni, Evan Pochtar, Tanmay Patwardhan

The Tokenizers

Problem Definition

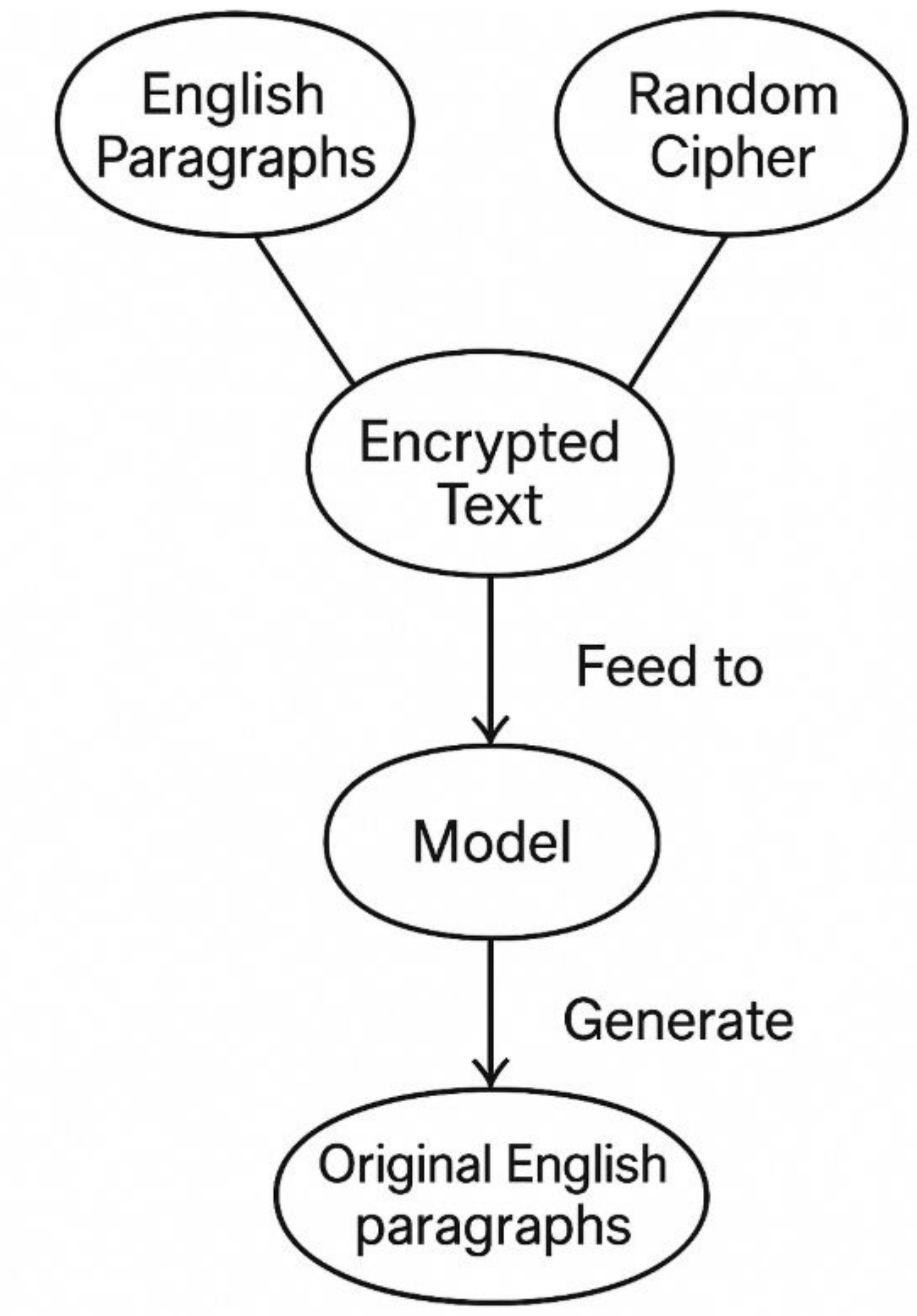
Our goal is to develop a small language model that can decrypt monoalphabetic substitution ciphers.

Motivation

By fine-tuning lightweight language models on cryptographic tasks, everyday users can decrypt ciphers on modest hardware, sidestepping the immense compute demands of large-scale LLMs.

Proposed Ideas

- Select a dataset of English text paragraphs.
- For each paragraph in the dataset, generate a random monoalphabetic cipher mapping each letter to another.
- Encrypt the dataset using these ciphers.
- Feed the encrypted text to the model and prompt it to decrypt the message into the original plaintext.



English Paragraphs

Random Cipher

Encrypted Text

Feed to

Model

Generate

Original English paragraphs

Literature Survey

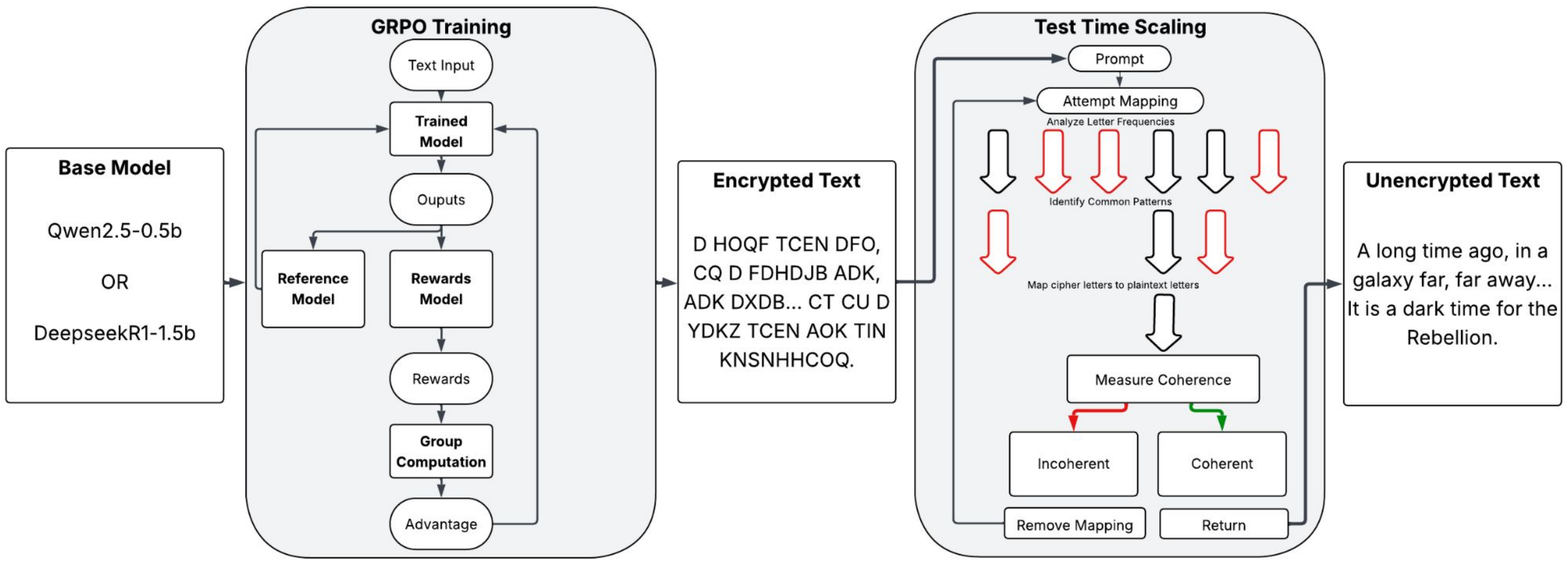
- Specialized prompt templates enhance ChatGPT's simple power analysis on cryptosystems (Zhou et al., 2024)
- FoC-BinLLM extracts semantic summaries from stripped binaries, handling tweaks and outperforming ChatGPT (Shang et al., 2024).
- Compression and fine-tuning optimize transformer models for resource-limited devices (Baek et al., 2024).
- **Summary:** Targeted prompts, semantic extraction, and architectural tweaks empower compact models to tackle complex cryptographic tasks, bolstering substitution cipher analysis.

Methodology

- Finetune a small language model using Group Relative Policy Optimization (GRPO) and use Test-Time Scaling for better inference results.
- GRPO training uses a reward function to score multiple outputs and adjust the model based on the reward of each output.
- Test-Time Scaling generates and scores multiple outputs during inference and selects the output with the highest reward to use.

Models Trained: We initially tried fine-tuning Qwen2.5-0.5B-Instruct but eventually moved on to fine-tuning Phi-4-mini-instruct when a larger model size was needed.

Training Method: For GRPO fine-tuning, we utilized the GRPOTrainer class from HuggingFace, and distributed the training across x4 A100 GPUs using MSI.



Base Model

Qwen2.5-0.5b

OR

DeepseekR1-1.5b

GRPO Training

Text Input

Trained Model

Ouputs

Reference Model

Rewards Model

Rewards

Group Computation

Advantage

Encrypted Text

D HOQF TCEN DFO, CQ D FDHDJB ADK, ADK DXDB... CT CU D YDKZ TCEN AOK TIN KNSNHHCOQ.

Test Time Scaling

Prompt

Attempt Mapping

Analyze Letter Frequencies

Identify Common Patterns

Map cipher letters to plaintext letters

Measure Coherence

Incoherent

Coherent

Remove Mapping

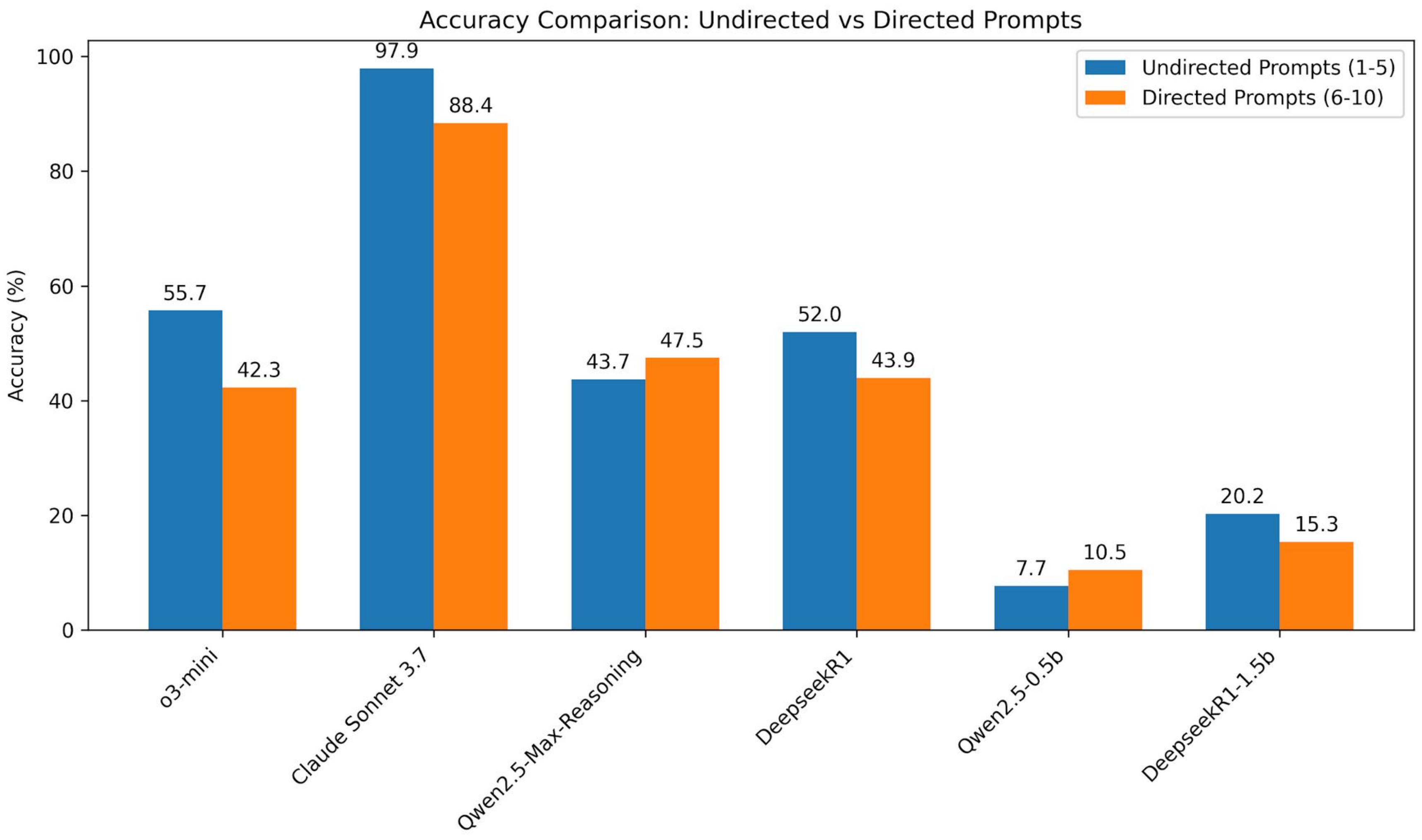
Return

Unencrypted Text

A long time ago, in a galaxy far, far away... It is a dark time for the Rebellion.

Results and Findings

- Most state-of-the-art models struggle (other than Claude interestingly).
- Experimenting with Directed vs Undirected prompting had mixed results.
  - In this context, directed prompts means that detailed step-by-step instructions were provided in the prompt, whereas undirected prompts simply asked the model to decipher the text.
- While our fine-tuned models performed better than the untrained model, it was still far from solving the cipher.



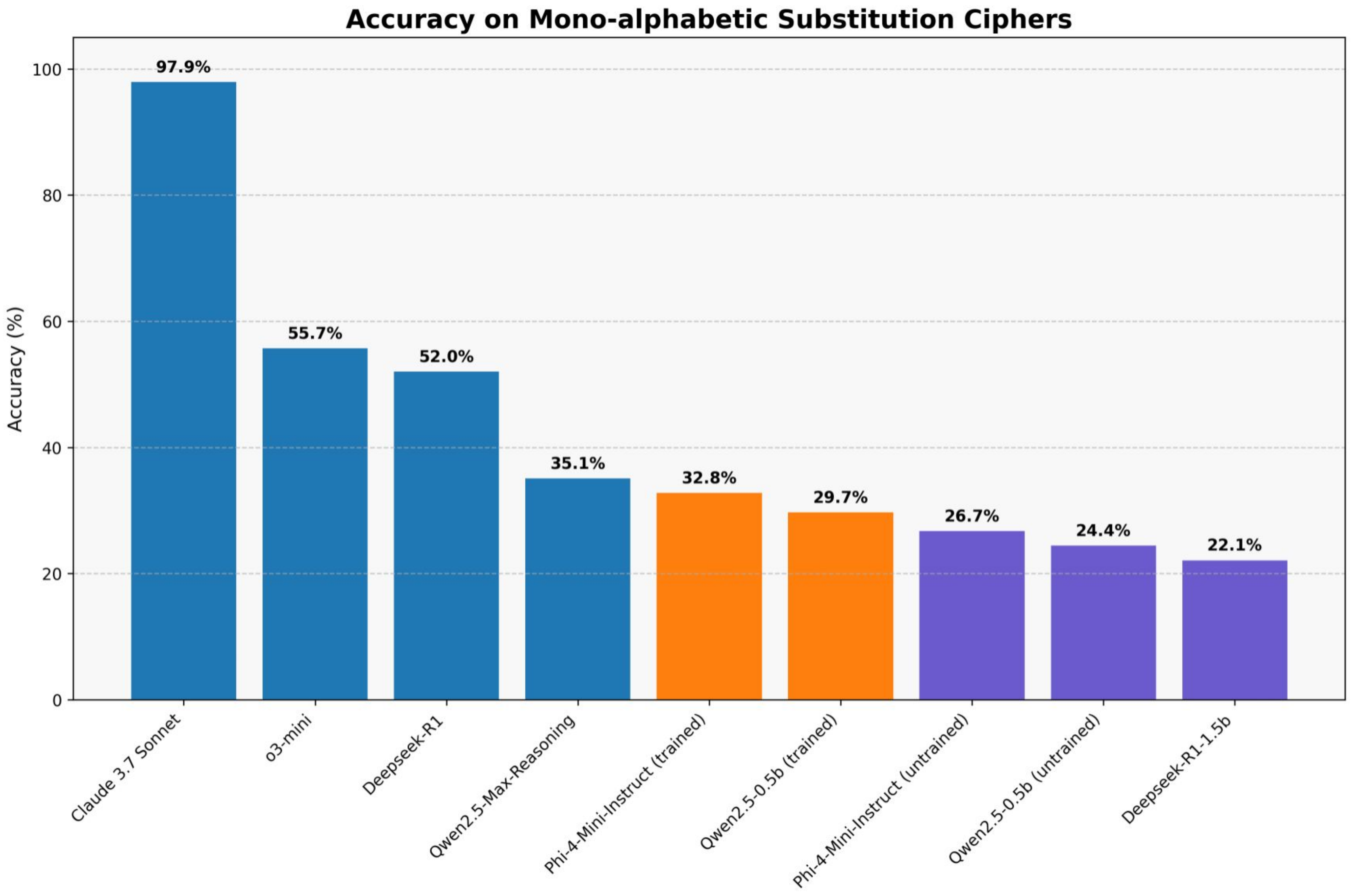
Accuracy Comparison: Undirected vs Directed Prompts

Limitations and Discussion

- Our current training uses 5000 data points that are encrypted, but using more data points could potentially increase accuracy.
- More work has to be done to investigate Claude 3.7 Sonnet's success in this field and the relative failure of other LLMs.
- Due to limited computational resources, larger base models were not available. However, in our investigations we observe that the accuracy increases as the base model parameters increase.

Contribution

- The main contribution of our project is that it demonstrates the difficulty of cryptographic tasks for LLMs, as well as shows that GRPO fine-tuning can lead to improvements over pretrained models on such tasks.



Accuracy on Mono-alphabetic Substitution Ciphers

Plan for the Final Report

- For the final report, we plan to do more analysis of our results and get to the root cause of why almost every LLM we tested struggles so much with this decryption task.