LLM Prompt-Tuning Framework for Responsible Student Use

Table of Content:

1. Objective

This study investigates how large language models (LLMs) handle structured algorithmic problems when faced with variation in **input quality, correctness, and educational target level**.

The framework below outlines how to design, categorize, and evaluate prompts in a reproducible way that can later be adapted for:

- **Other algorithm types** (sorting, dynamic programming, recursion)
- **Other educational subjects** (mathematics, physics, language learning)
- **Data management applications** (handling incomplete or noisy data inputs)

2. Prompt Design Dimensions

Each prompt is categorized according to **five key tuning parameters** that reflect real-world input variability:

Parameter	Description	Educational/Data Analogy	Evaluation Metric
Cleanliness	Measures understanding and correctness despite how well-formed or noisy a prompt is.	Clean = structured teacher question Noisy = student's fragmented or typo-ridden query.	Model detects true intent of question (Yes/No or multiple choice) and gives correct answer. Measure on correctness.
Completeness	Determines whether all necessary information is given.	Like a partial dataset. Does the model recognize there is missing information? Does it infer the missing pieces?	Accuracy of inference and acknowledgment of missing data.
Confidently Wrong	Tests whether a model corrects a *confidently incorrect input*.	Mimics student misunderstandings. Will a model correct even if the user prompts a wrong input.	Model must explicitly identify the misconception and correct it. Measured on correctness.
Explanation Depth	Measures quality and completeness of conceptual explanation.	Graded by a "checklist" depending on the algorithm and level.	Binary/Partial match to checklist requirements. Additional optional human evaluation.
Target Audience	Adjusts readability and conceptual framing to a student grade level.	K–12 and university learning adaptation.	Flesch–Kincaid, Gunning Fog, and Type-Token Ratio.

Each task (BFS, DFS, Cycle Detection, Topological Sort, Tarjan's) increases in *cognitive and algorithmic difficulty* while maintaining the same parameter categories, enabling clean comparison across difficulty levels.

3. Example Framework for Prompt Creation

Task	Difficulty	Input Variation	Target Audience	Expected Model Behavior
BFS	Easy	Clean	Grade 5	Short analogy with correct order.
BFS	Easy	Noisy	Grade 8	Detect intent despite typos.
BFS	Easy	Confidently Wrong	Grade 9	Identify BFS vs DFS error.
DFS	Medium	Incomplete	Grade 10	Infer missing graph edges, maintain logic.
Topological Sort	Medium	Confidently Wrong	Grade 8	Reject claim that topo sort works on cyclic graph.
Tarjan's SCC	Hard	University	University	Present structured explanation and complexity proof.

4. Adaptation and Reproduction Guide

To apply this framework to other domains, follow these steps:

1. **Choose a Concept Family:**
   e.g., recursion, sorting algorithms, probability, grammar correction.

2. **Design a 5-level difficulty curve:**
   From a foundational example (BFS) to an advanced, abstract one (Tarjan's).

3. **Vary Prompt Input Quality:**

   - Add noise (misspellings, redundant text, incomplete data).

   - Introduce misconceptions intentionally.

   - Remove or modify key context clues.

4. **Target Different Learners:**
   Introduce tasks that require certain grade level understanding in a range for **elementary, high school, and university** readers.
   This creates "educational adaptability" metrics.

5. **Clean then apply the Evaluation Pipeline:**
   Apply python response cleaning script to normalize long written responses. Then use the readability, correctness, and error-detection tests from Python code.
   Compare results per model and per difficulty level.

5. Data Management Perspective

This experiment aligns with data management problems because:

- It simulates **incomplete, inconsistent, or noisy inputs**, like real-world datasets.
- It tests **model robustness**. How reliably a LLM can produce consistent results under data variation.
- It contributes to **data cleaning and understanding pipelines**, showing that prompt clarity and structure are part of "data quality."