Winning Countdown: Surrogate Models for Efficient and Interpretable Prompt Engineering

Machine Learning Under a Modern Optimization Lens

Gabriel Yong, Lara Pontes

MIT

December 10, 2025

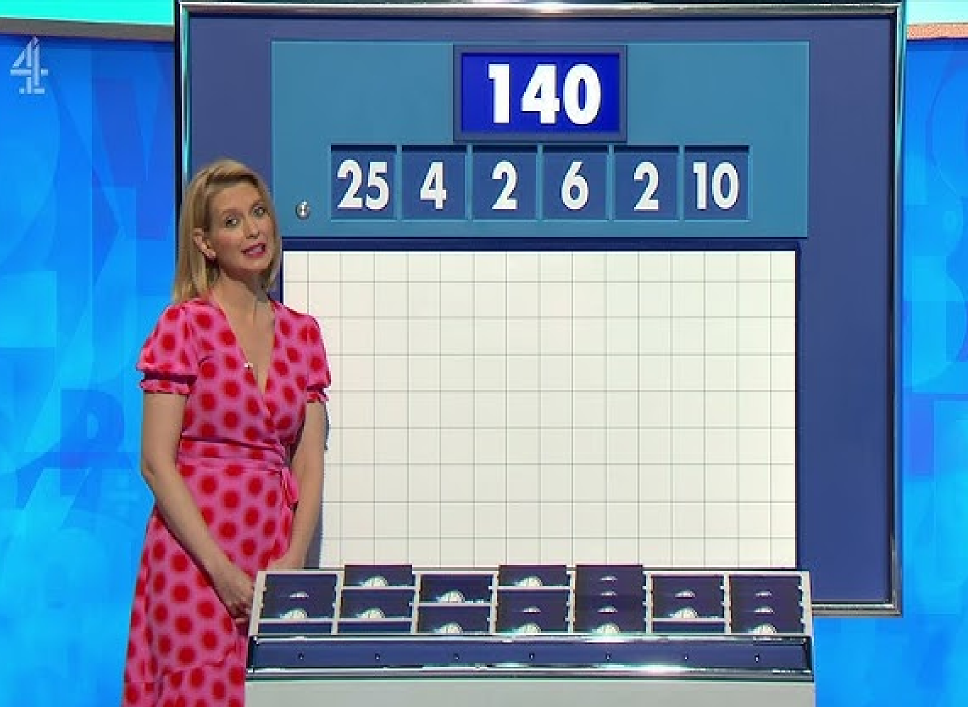
Outline

Outline

The Challenge: Prompt Engineering is Expensive

Motivation

- LLMs are highly sensitive to how tasks are phrased [3]
- Prompt engineering is essential for accurate performance
- **Problem:** Evaluating every possible prompt on an LLM is prohibitively expensive
- **Question:** Can we predict if an LLM will produce a correct answer without actually running it?
- **Research opportunity:** New prompt-engineering methods and interpretable models to refine the interface between human and machine-level prompting


140
25
4
2
6
2
10

Problem Formulation

Setup

- Let I = set of prompt templates (how to explain the countdown task)
- Let J = set of task instances (the list of numbers and target)
- For each (i, j) pair: $Y_{ij} \in \{0, 1\}$ indicates correctness

Goal

Learn a function f such that:

$$f(x_i, z_j, P_{ij}) \approx Y_{ij}$$

where:

- x_i = prompt features (e.g., prompting strategies)
- z_j = instance features (e.g., complexity metrics)
- P_{ij} = combined textual input

Data

Task Description

"Given a list of numbers and a target value, use each number exactly once with basic arithmetic operations $(+, -, *, /)$ to reach the target."

Dataset

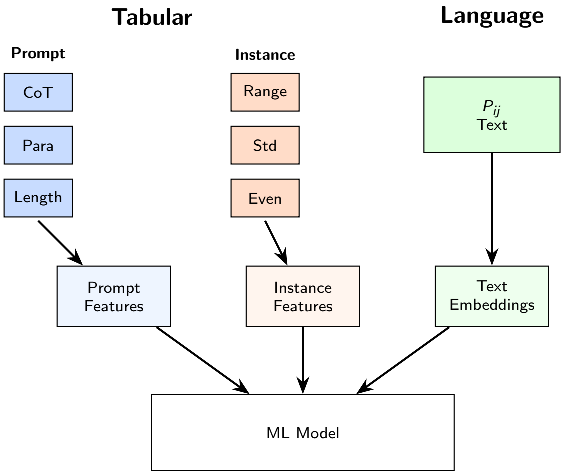
- Nearly **6,000** diverse prompt templates generated using Gemini 2.5 Flash
- Strategies: paraphrasing, role-specification, reasoning-trigger, chain-of-thought, self-check, conciseness, verbosity, context-expansion, few-shot-count
- **2,000** countdown instances from public dataset
- Prompted Gemini 2.5 Flash-Lite to get more than **11,000** sample responses with at most 2,048 tokens ($\sim$ 0.1%)

Outline

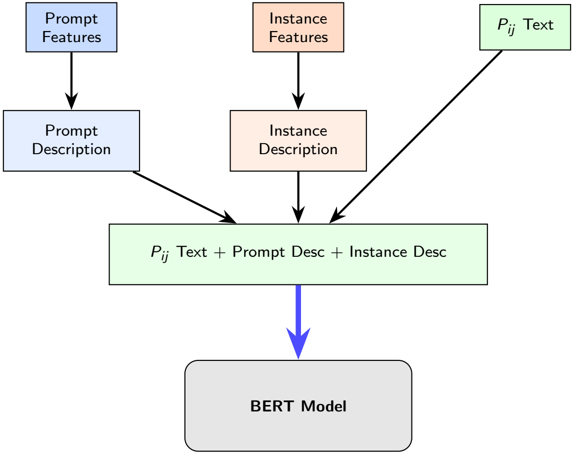
Models Benchmarked

- Logistic Regression
- Decision Tree
- Random Forest
- XGBoost
- Multi-Layer Perceptrons (MLPs)
- BERT
  - Pre-trained transformer model using ModernBERT for longer context length
  - Only trained with P_{ij} and natural language versions of x_i and z_j

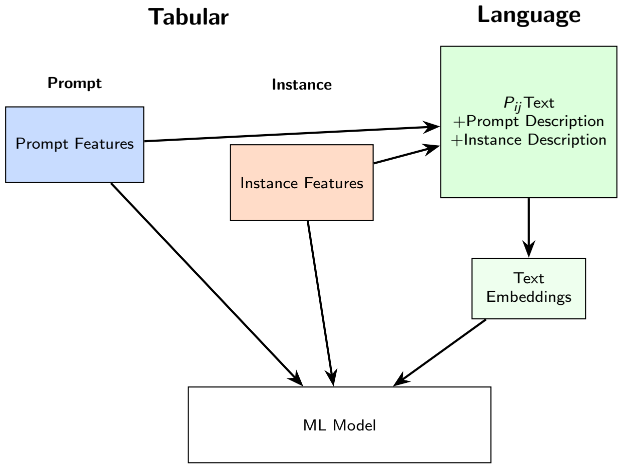
Model Training Pipeline



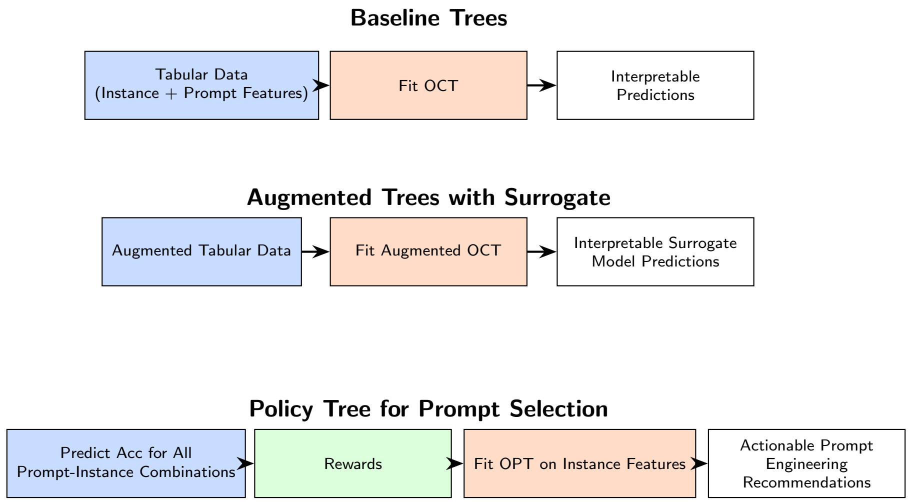
BERT Model Pipeline



Model Training Pipeline with Natural Language Descriptions



Tree Pipeline



Baseline Trees
Tabular Data (Instance + Prompt Features)
Fit OCT
Interpretable Predictions
Augmented Trees with Surrogate
Augmented Tabular Data
Fit Augmented OCT
Interpretable Surrogate Model Predictions
Policy Tree for Prompt Selection
Predict Acc for All Prompt-Instance Combinations
Rewards
Fit OPT on Instance Features
Actionable Prompt Engineering Recommendations

Outline

Model F1

Model	all	inst+prompt	inst	prompt	text	text+prompt
	x_i, z_j, P_{ij}	x_i, z_j	z_j	x_i	P_{ij}	x_i, P_{ij}
lr	0.7606	0.775	0.7125	0.5386	0.6761	0.687
dt	0.7548	0.7515	0.6951	0.5481	0.5443	0.5477
rf	0.7644	0.7807	0.7082	0.5471	0.5716	0.586
xgboost	**0.7948**	**0.7859**	0.7077	0.5389	0.604	0.5973
mlp	0.7748	0.7817	**0.7223**	**0.5964**	0.6961	0.6854
bert	0.7787	0.5771	0.0	0.5396	**0.7746**	**0.7699**
oct	-	0.7656	-	-	-	-
oct (augmented)	-	0.7734	-	-	-	-

Table: Test F1 by Model and Feature Mode (w/out NL features)

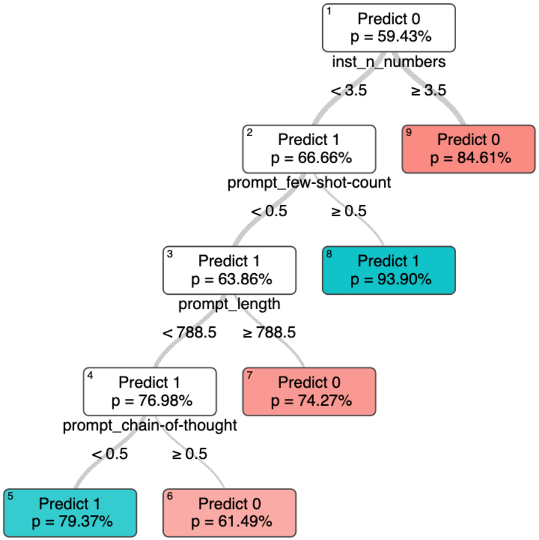
Model	all	inst+prompt	inst	prompt	text+prompt
	x_i, z_j, P_{ij}	x_i, z_j	z_j	x_i	x_i, P_{ij}
lr	0.7601	0.7741	0.7022	0.5321	0.6746
dt	0.746	0.7364	0.6816	0.4698	0.5405
rf	0.75	0.7641	0.6833	0.5391	0.5701
xgboost	**0.7873**	**0.7848**	0.6919	0.5331	0.5648
ann	0.7838	0.7786	**0.7253**	**0.6466**	0.6993
bert	0.7787	0.5771	0.0	0.5396	**0.7699**

Table: Test F1 by Model and Feature Mode (w/ NL features)

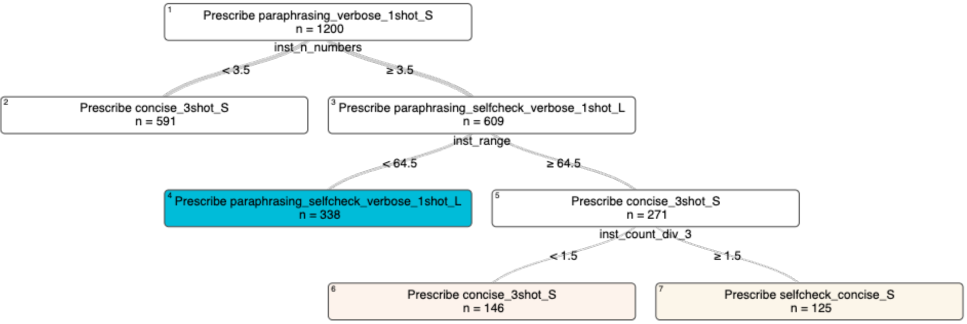
Feature Importance Analysis

- **Instance features dominate**
  - Strong predictors of correctness across models
- **Prompt features alone are weak**
  - Performance drops below 60% F1 without instance information
  - Text embeddings help but remain limited
- **BERT uses text most effectively**
  - Best model when only prompt information is available ($\sim$78% F1)
  - Captures semantic nuances other models miss

OCT: Human-Level Interpretability



OPT: From Predictions to Prescriptions


1 Prescribe paraphrasing_verbose_1shot_S
n = 1200
inst_n_numbers
< 3.5
≥ 3.5
2 Prescribe concise_3shot_S
n = 591
3 Prescribe paraphrasing_selfcheck_verbose_1shot_L
n = 609
inst_range
< 64.5
≥ 64.5
4 Prescribe paraphrasing_selfcheck_verbose_1shot_L
n = 338
5 Prescribe concise_3shot_S
n = 271
inst_count_div_3
< 1.5
≥ 1.5
6 Prescribe concise_3shot_S
n = 146
7 Prescribe selfcheck_concise_S
n = 125

- Mean reward using best single template on the test set: 0.7527
- Mean reward using learned policy on the test set: 0.7799

Outline

Key Insights

- **Surrogate models as efficient approximators**
    - Predict LLM correctness with $> 80\%$ accuracy
    - Dramatically reduce expensive LLM evaluations by filtering poor prompts early
- **Trees improve interpretability without losing performance**
    - OCTs bridge the gap between machine and human understanding of task-specific prompt engineering
    - OPTs provide actionable recommendations based on instance features
- **Practical optimization pipeline for prompt engineering**
    - Learn a surrogate $\rightarrow$ estimate rewards $\rightarrow$ train an OPT

Thank you! Questions?

Literature

Surrogate Modeling

- Polo et al. [2]: Efficient evaluation of LLMs on a task over different prompts via statistical summaries from random samples
- Kadavath et al. [1]: Train models to estimate the probability of generating a true answer, but rely on internal logit-based signals (requires access to the LLM, not possible for proprietary models)

Prompt Template Generation and Features (x_i)

Generation Instruction

"Given an original prompt, create exactly 5 alternative prompts that preserve the same underlying task as the original prompt, but may vary phrasing, structure, and additional guidance. Do not change what the original prompt is ultimately asking the model to do."

Prompt Features

- **Paraphrasing**
- **Role specification:** "As an **expert in mathematical puzzles**..."
- **Reasoning trigger:** "Think step by step..."
- **Chain-of-thought:** "Outline your strategy, explore various combinations..."
- **Self-check:** "Verify correctness before producing the final answer."
- **Conciseness**
- **Context expansion:** "Imagine you're in the game show Countdown..."
- **Length of the prompt** (number of characters)
- **Few-shot count** (number of examples)

Instance Features (z_j)

Feature Categories

- **Basic Numerical Structure**
  - Count of integers in the instance
  - Range (max − min)
  - Standard deviation
- **Number Properties**
  - Count of small/large numbers (< 10 or > 50)
  - Count of duplicate numbers
  - Parity and divisibility (`count_even`, `count_odd`, `count_div_2/3/5/7`)
  - Count of prime numbers
- **Difficulty Heuristics**
  - Count of "clean" pair operations (exact division, sum ending in 0)
- **Solution-derived Features** (Only known if solution is known)
  - Depth of the expression tree
  - Operator counts: `count_add`, `count_sub`, `count_mul`, `count_div`
  - Number of noncommutative operations (subtractions/divisions)

Dataset Construction Pipeline

Template + Instance = Filled Prompt

- **5,813** prompt templates with varying styles
- **1,325** unique number instances
- **2,241** prompt-instance pairs in test set

Model Accuracy

Model	all	inst+prompt	inst	prompt	text	text+prompt
	x_i, z_j, P_{ij}	x_i, z_j	z_j	x_i	P_{ij}	x_i, P_{ij}
lr	0.809	0.8215	0.7461	0.6506	0.7452	0.7532
dt	0.8014	0.8014	0.7421	0.6475	0.6234	0.6109
rf	0.8215	0.83	0.751	0.6484	0.6769	0.6885
xgboost	**0.8389**	**0.8322**	0.7497	**0.6586**	0.6992	0.689
mlp	0.8166	0.8273	**0.755**	0.6479	0.7421	0.7349
bert	0.822	0.4056	0.5944	0.6497	**0.8273**	**0.8237**
oct	-	0.8210	-	-	-	-
oct (augmented)	-	0.8144	-	-	-	-

Table: Test Accuracy by Model and Feature Mode (w/out NL features)

Model	all	inst+prompt	inst	prompt	text+prompt
	x_i, z_j, P_{ij}	x_i, z_j	z_j	x_i	x_i, P_{ij}
lr	0.8099	0.8224	0.7456	0.6452	0.7443
dt	0.7934	0.7831	0.7278	0.6203	0.6252
rf	0.8135	0.8206	0.7332	0.6475	0.6756
xgboost	**0.8353**	**0.8309**	0.7354	0.6444	0.6747
ann	0.8264	0.8117	**0.7506**	0.6122	0.7452
bert	0.822	0.4056	0.5944	**0.6497**	**0.8237**

Table: Test Accuracy by Model and Feature Mode (w/ NL features)

Example 1: Simple Concise Prompt

Step 1: Prompt Template (ID: 1047)

Generate an equation using the numbers `<numbers_placeholder>` to reach the target `<target_placeholder>`. You may use +, -, *, / operations. Each number from the given set must be used one and only one time.
After constructing your equation, critically review it to ensure it meets all conditions and correctly evaluates to `<target_placeholder>`. If necessary, make corrections.
The final, verified equation should be placed inside `<answer></answer>` tags. Any thought process or derivation steps should go into `<think></think>` tags. Do not include any text outside these tags.

Template Features: paraphrasing=1, self-check=1, length=574 chars

Natural Language Description: "This prompt uses paraphrasing, self-checking requests. The prompt includes no few-shot examples. The prompt has a total length of 574 characters."

Example 1: Simple Concise Prompt (continued)

Step 2: Instance (ID: 239)

Numbers: [49, 44, 1]
Target: 94

Instance Features:

- Range: 48, Standard Deviation: 26.39
- Even count: 1, Odd count: 2
- Distance to target (simple ops): 0.0

Natural Language Description: "This problem involves 3 numbers with a range of 48.0 and a standard deviation of 26.39. The distribution includes 1 even numbers and 2 odd numbers..."

Example 1: Simple Concise Prompt (continued)

Step 3: Filled Prompt (Template + Instance)

Generate an equation using the numbers **[49 44 1]** to reach the target **94**. You may use +, -, *, / operations. Each number from the given set must be used one and only one time.
After constructing your equation, critically review it to ensure it meets all conditions and correctly evaluates to **94**. If necessary, make corrections.
The final, verified equation should be placed inside `<answer></answer>` tags. Any thought process or derivation steps should go into `<think></think>` tags. Do not include any text outside these tags.

This filled prompt is sent to the LLM for evaluation.
Solution: $49 + 44 + 1 = 94$

Example 2: Expert Role + Chain-of-Thought

Step 1: Prompt Template (ID: 4101)

You are a master mathematician tasked with solving a numerical puzzle. Your objective is to meticulously construct an equation that precisely evaluates to `<target_placeholder>`, utilizing every number from the set `<numbers_placeholder>` exactly one time. You are restricted to using only the fundamental arithmetic operations: addition (+), subtraction (-), multiplication (*), and division (/).
Before presenting your final solution, ...

Features: role-specification=1, chain-of-thought=1, verbosity=1, length=858 chars

Natural Language Description: "This prompt uses role specification, chain-of-thought prompting, verbosity requirements. The prompt includes no few-shot examples. The prompt has a total length of 858 characters."

Example 2: Expert Role + Chain-of-Thought (continued)

Step 2: Instance (ID: 1919)

Numbers: [28, 57, 73]
Target: 12

Instance Features:

- Range: 45, Standard Deviation: 22.5
- All odd numbers (odd count: 3)
- More challenging - requires subtraction/division

Example 2: Expert Role + Chain-of-Thought (continued)

Step 3: Filled Prompt

You are a master mathematician tasked with solving a numerical puzzle. Your objective is to meticulously construct an equation that precisely evaluates to **12**, utilizing every number from the set **[28 57 73]** exactly one time. You are restricted to using only the fundamental arithmetic operations: addition (+), subtraction (-), multiplication (*), and division (/).
Before presenting your final solution, you must demonstrate your full problem-solving process. Think step-by-step, exploring different combinations and intermediate calculations. Document all your reasoning, attempts, and derivations within `<think></think>` tags. Once you have confidently arrived at the correct equation, present it within `<answer></answer>` tags. No other text is permitted outside these tags.

Prompt Style Features

The 5,813 templates vary across 8 stylistic dimensions:

- **Paraphrasing** (88.6%)
- **Role-specification** (21.6%)
- **Chain-of-thought** (20.9%)
- **Self-check** (20.2%)
- **Few-Shot** (10.1%)
- **Reasoning-trigger** (33.9%)
- **Conciseness** (30.7%)
- **Verbosity** (27.3%)
- **Context-expansion** (19.8%)

Length Range: 35 - 1,958 characters (avg: 652)

Instance Characteristics

Each of the 1,325 unique instances contains:

Core Elements:

- 3-4 numbers and a target value
- Example: Numbers=[49, 44, 1], Target=94

25 Numerical Features:

- Statistical: range, standard deviation
- Composition: even/odd counts, divisibility properties
- Difficulty: distance to target, expression depth
- Operations: count of add/sub/mul/div in solution

Natural Language Description:

- Human-readable explanation of instance properties
- Used in `_w_nl` feature modes

Feature Modes for Prediction

We train models on **11 different feature combinations**:

Base Modes (6):

1. `text_only` - Only prompt text (BERT embeddings)
2. `features_only` - All numeric features
3. `inst_only` - Only instance features
4. `prompt_only` - Only prompt features
5. `text_prompt` - Text + all numeric features
6. `all` - Everything combined

Extended with Natural Language (5):

7. `inst_w_nl`, `prompt_w_nl`, `features_w_nl`
8. `text_prompt_w_nl`, `all_w_nl`

Model Training Overview

6 Model Types:

- Logistic Regression, Random Forest, Decision Tree
- XGBoost, ANN (Artificial Neural Network), BERT

Training Configuration:

- 5 models $\times$ 11 modes $+$ BERT $\times$ 6 modes $=$ **61 configurations**
- Train/Val/Test split from diverse sampling strategies
- SLURM job arrays for distributed training

Prediction Goal:

- Binary classification: Will the LLM solve correctly?
- Current baseline: 40.6% of problems solved correctly
- Research question: Which features best predict LLM success?

Key Innovations

1. **Multi-modal Feature Engineering**
   - Combines text (BERT), numeric, and NL features
   - Systematic exploration of feature combinations
2. **Natural Language Feature Descriptions**
   - Human-readable explanations of numeric features
   - Leverages LLM's language understanding
3. **Systematic Prompt Engineering Analysis**
   - 5,813 templates with controlled variations
   - Quantifies impact of prompt style on LLM performance
4. **Comprehensive Model Comparison**
   - Classical ML vs. Deep Learning approaches
   - 61 model-feature configurations

Saurav Kadavath et al.
Language models (mostly) know what they know.
arXiv preprint arXiv:2207.05221, 2022.

Maia Polo et al.
Efficient multi-prompt evaluation of llms.
arXiv preprint arXiv:2405.17202, 2024.

Daniel Ye et al.
How predictable are large language model capabilities? a case study on big-bench.
arXiv preprint arXiv:2305.15701, 2023.