An In-Depth Analysis of Microsoft's PromptWizard Ecosystem for the Development of a Commercial Prompt Enhancement Model

Deconstruction of Microsoft's PromptWizard Framework

An examination of Microsoft's PromptWizard repository reveals that it is not a model to be trained or a library for direct integration into production applications, but rather a sophisticated, task-aware, agent-driven framework for discrete prompt optimization.[1] Its core purpose is to automate the labor-intensive process of prompt engineering by using a Large Language Model (LLM) as an autonomous agent to discover highly effective prompts for a given task. This process operates in the "black-box" setting, meaning it does not require access to the target LLM's internal weights or gradients, relying solely on API-based interactions.[1] This makes the framework versatile, capable of optimizing prompts for proprietary models like those from OpenAI or Microsoft Azure.

The fundamental architecture of PromptWizard is a self-evolving mechanism designed to iteratively generate, critique, and refine both the instructional components of a prompt and its in-context learning examples.[1] This methodology is positioned as a highly efficient alternative to manual prompt design, with Microsoft's research indicating a significant reduction in computational overhead, costing as little as $0.05 per task and decreasing token consumption by a factor of 5 to 60 compared to other automated methods.[2]

This framework is best understood not as a real-time engine for transforming user input, but as a design-time tool for discovering a single, highly optimized "meta-prompt" or template. The iterative, multi-call nature of its optimization loop, while cost-effective for development, would introduce prohibitive latency and expense in a live, user-facing commercial SaaS product. Therefore, the value of PromptWizard for such an application lies in its principles and methodologies, which can guide the creation of a specialized dataset to fine-tune a dedicated, low-latency prompt enhancement model, rather than in the direct deployment of the framework itself.

Core Architecture: Task-Aware Agent-driven Optimization

The central thesis of PromptWizard is that an LLM can be guided to systematically improve its own instructions through a structured, feedback-driven process.[1] It treats prompt optimization as a discrete search problem within the vast space of possible text sequences. The "agent" in this context is the LLM itself, which is directed by the framework's logic to perform a series of actions—generation, mutation, evaluation, critique, and synthesis—to navigate this space and converge on a high-performing prompt.

This self-adaptive approach ensures a holistic optimization strategy. It does not merely tweak the instructional text in isolation; it co-evolves the instructions and the in-context learning (few-shot) examples, recognizing that the interplay between these two components is critical for achieving state-of-the-art performance.[2] By continuously refining both elements in tandem, the framework balances exploration of new prompt variations with the exploitation of proven high-performing structures.

The Generate-Critique-Refine Loop: A Self-Evolving Mechanism

The engine of PromptWizard is an iterative feedback loop that continuously improves prompt components. This process can be broken down into two parallel, yet interconnected, optimization tracks for instructions and examples.

Instruction Optimization: The process begins with a user-provided base_instruction and a task_description.[1] The framework then prompts a powerful LLM to "mutate" this initial instruction, generating several variations using cognitive heuristics.[3] These candidate instructions are then embedded into a complete prompt structure and evaluated against a small, curated set of training examples (defined by the seen_set_size parameter). The performance of each variation is measured, and the best-performing prompt is selected as the new baseline for the next iteration.

In-Context Example Optimization: Concurrently, PromptWizard optimizes the few-shot examples that provide context to the LLM. It begins by selecting a diverse set of examples from the provided training data. The framework then identifies "positive" and "negative" examples based on their performance with the current best prompt instruction.[1] The negative examples—those where the model failed to produce the correct answer—are particularly valuable. The "critique" component analyzes these failures to understand *why* the prompt was insufficient. This feedback is then used in the "synthesize" step to generate new, synthetic examples that are more robust, diverse, and task-relevant, specifically designed to address the weaknesses identified in the critique phase.[1]

Feedback-Driven Synthesis: The critique and synthesis steps are the core of the framework's intelligence. After evaluating a set of prompt variations, the LLM is prompted to act as a critic, analyzing the strengths and weaknesses of the best-performing prompt and identifying gaps that need to be addressed.[2] This structured feedback is then provided to a

"synthesizer" module (another LLM prompt), which leverages the critique to generate a new, refined prompt instruction. This cycle of generation, testing, critique, and synthesis allows the prompt to evolve over multiple iterations, progressively improving its clarity, specificity, and effectiveness.

Operational Scenarios and Use Cases

The PromptWizard framework is designed with the flexibility to operate in three distinct scenarios, catering to different levels of available data and optimization goals [1]:

1. **Scenario 1: Optimizing Prompts without Examples (Zero-Shot).** In this mode, the framework focuses exclusively on refining the instructional text of the prompt. The optimization loop generates and tests variations of the instruction without any in-context examples, making it suitable for tasks where the goal is to create a powerful zero-shot prompt or where example data is unavailable.
2. **Scenario 2: Generating Synthetic Examples.** This scenario starts with a base prompt instruction and uses the framework's synthesis capabilities to generate a set of high-quality, synthetic few-shot examples. This is particularly useful for improving the in-context learning performance of a model when a large training dataset is not available for selection.
3. **Scenario 3: Optimizing Prompts with Training Data.** This is the most comprehensive and powerful use case. The framework leverages a small set of provided training data to iteratively optimize both the prompt instructions and the in-context examples in tandem. This holistic approach allows the instructions and examples to co-evolve, leading to the highest possible task performance.

Technical Implementation and Configuration

The practical application of the PromptWizard framework is governed by a specific data structure and a central configuration file. Understanding these components is essential for preparing data and controlling the optimization process, and provides a blueprint for creating a dataset for fine-tuning a custom prompt enhancement model.

Dataset Structure Requirements

PromptWizard mandates a simple yet specific format for all training and test data. This consistency ensures that the framework can reliably parse inputs and evaluate outputs during the optimization loop.

- **File Format:** All datasets must be provided in the **JSON Lines (.jsonl)** format.[1] In this format, each line of the file is a self-contained, valid JSON object, which allows for

efficient, streamable processing of large datasets.

- **Required Fields:** Each JSON object (i.e., each line in the file) must contain two mandatory key-value pairs [1]:
    - "question": A string that holds the complete input, query, or problem statement to be presented to the LLM.
    - "answer": A string containing the corresponding ground-truth solution. The framework is flexible regarding the content of this field; it can be a concise, single-value answer (e.g., a number or a class label) or a verbose, multi-step explanation that includes the reasoning process.

An example of a valid data instance, contextualized for the GSM8K dataset frequently referenced by the repository, is as follows:

JSON

```
{
  "question": "Natalia sold clips to 48 of her friends in April, and then she sold half as many clips in May. How many clips did Natalia sell altogether in April and May?",
  "answer": "Natalia sold 48/2 = <<48/2=24>>24 clips in May.\nNatalia sold 48+24 = <<48+24=72>>72 clips altogether in April and May.\n#### 72"
}
```

This structure is derived from the descriptions in [1] and.[5]

The Control Plane: promptopt_config.yaml

The promptopt_config.yaml file serves as the central control panel for the entire optimization process.[1] It allows the user to define the task, set the initial conditions for the prompt, and tune the parameters of the search algorithm. The key parameters within this file have a direct and strategic impact on the outcome of the optimization.

Parameter	Function & Strategic Impact	Data Type	Example Value / Recommended Range	Source
task_description	Sets the high-level context and persona for the LLM. This is crucial for role-prompting and aligning the	String	"You are a mathematics expert. You will be given a mathematics problem which you need to	[1]

	model with the desired domain expertise.		solve."	
base_instruction	The initial instruction that serves as the starting point for the evolutionary optimization process. A well-chosen base can significantly accelerate convergence.	String	"Lets think step by step."	[1]
answer_format	A critical instruction that constrains the LLM's output format, enabling reliable, automated extraction of the final answer for evaluation. Mismatches here lead to evaluation failure.	String	"At the end, wrap only your final option between <ANS_START> and <ANS_END> tags."	[1]
seen_set_size	Determines the number of training samples used to evaluate each prompt variation. This parameter balances the strength of the performance signal against the computational cost and API usage of the optimization run.	Integer	20-50 (Microsoft used 25 in experiments)	[1]
few_shot_count	Specifies the	Integer	Any positive	[1]

	number of in-context examples to be included in the prompt. This can be set to a small integer (e.g., 2-5) for few-shot optimization or adjusted for discovering zero-shot prompts.		integer (e.g., 2-5)	

The Role of Chain-of-Thought (CoT) Reasoning

PromptWizard explicitly incorporates and optimizes for Chain-of-Thought (CoT) reasoning, a technique that significantly enhances an LLM's ability to solve complex, multi-step problems.[1] The framework's ability to generate detailed reasoning chains is a key feature that enriches the problem-solving capacity of the final optimized prompt.
The default base_instruction suggested in the repository's examples, "Let's think step by step," is a direct and well-documented invocation of **Zero-Shot-CoT**.[1] This simple phrase is known to elicit a step-by-step reasoning process from capable LLMs, prompting them to "show their work" before arriving at a final answer. This aligns perfectly with the structure of benchmark datasets like GSM8K, where the ground-truth
answer field contains not just the final solution but also the intermediate logical and mathematical steps required to reach it.[5] By optimizing prompts that encourage this behavior, PromptWizard produces instructions that are not only more accurate but also more transparent and auditable.

Training Methodologies: Beyond Conventional Fine-Tuning

A crucial distinction to make is that PromptWizard is a framework for *prompt discovery*, not *model training*. It operates on the principle of discrete prompt optimization, which stands in contrast to methods that involve updating the weights of an LLM, such as full fine-tuning or parameter-efficient techniques like LoRA. Understanding this distinction is vital for placing PromptWizard within the broader landscape of LLM optimization and for selecting the appropriate methodology for building a dedicated prompt enhancement model.

Discrete Prompt Optimization: The Core of PromptWizard

The methodology at the heart of PromptWizard is a search algorithm that navigates the discrete space of text tokens to find an optimal prompt sequence.[1] No model parameters are ever modified during this process; the underlying LLM is treated as a fixed, black-box function.[1] The "learning" occurs within the prompt itself, which is iteratively refined based on performance feedback from a small evaluation dataset. This approach can be conceptualized as a form of automated, agentic in-context learning, where the framework discovers the best way to instruct the model for a specific task without altering the model's fundamental knowledge or capabilities. Its primary advantage is its accessibility and efficiency; it can be applied to any model with an API, and it requires only a small number of training examples to guide its search.[1]

Comparative Analysis: Alternative Prompt Optimization Paradigms

While PromptWizard offers a powerful method for prompt discovery, it is one of several paradigms for automatic prompt engineering. Other approaches, particularly those referenced in related Microsoft research, involve more intensive training and result in fundamentally different types of artifacts.

- **Reinforcement Learning (RL):** Microsoft's LMOps repository and associated research on a system called **Promptist** showcase an RL-based approach to prompt optimization, particularly for the text-to-image domain.[6] In this paradigm, a separate language model is trained to act as a "prompt policy network." This network takes a user's initial, simple prompt and rewrites it. The rewritten prompt is then used to generate an image, and a reward function—which typically combines an aesthetic quality score and a semantic relevance score (e.g., CLIP score)—evaluates the result.[7] This reward signal is used to update the policy network via RL algorithms, teaching it to generate prompts that produce better images over time. This is a model-centric approach that is more computationally intensive than PromptWizard's search but can result in a highly adaptive prompt rewriting model.
- **Instruction Evolution:** The methodology behind the highly successful **WizardLM** family of models is known as **Evol-Instruct**.[9] This is a data-centric approach focused on enhancing the quality and complexity of an entire *dataset* of instructions. Starting with a small seed set of simple instructions, an LLM is used to progressively rewrite them to be more complex through a variety of mutation operations, such as adding constraints, deepening the required reasoning, concretizing abstract concepts, or increasing the reasoning steps.[10] The goal of Evol-Instruct is not to find a single optimal prompt for one task, but to bootstrap a large-scale, high-quality instruction-following dataset. This evolved dataset is then used to conduct supervised

fine-tuning on a base LLM, thereby instilling more advanced instruction-following capabilities into the model's weights.[9]

The existence of these distinct methodologies reveals a spectrum of automation in prompt engineering. On one end, PromptWizard provides a lightweight, API-driven framework for discovering a static prompt template. In the middle, Evol-Instruct offers a powerful data-synthesis pipeline for creating superior fine-tuning datasets. At the most complex end, Promptist demonstrates how a dedicated model can be trained with RL to dynamically rewrite prompts. For the objective of creating a production-grade Qwen3-30B model that transforms vague user inputs, the Evol-Instruct paradigm is the most directly applicable strategy. It provides a clear path for creating the necessary "vague prompt" → "enhanced prompt" dataset required for supervised fine-tuning.

Technique	Methodology	Primary Goal	Data Requirement	Computational Cost	Ideal Use Case
PromptWizard	Discrete Search (Agent-driven Critique & Refine)	Discover a single, optimal static prompt for a specific task.	Small set of examples (20-50) for evaluation.	Low (API calls only, ~$0.05/task).	Quickly finding a high-performing prompt for a well-defined task without model training.
Evol-Instruct (WizardLM)	Instruction Dataset Evolution (LLM-driven rewriting)	Create a large-scale, high-complexity instruction dataset for fine-tuning.	Small seed set of initial instructions.	Medium (significant API calls for data generation, plus cost of fine-tuning).	Upgrading the core instruction-following capability of a base LLM.
Promptist (RL)	Reinforcement Learning (training a policy model)	Train a dedicated model that dynamically rewrites user prompts to maximize a reward signal.	Small set of engineered prompts for initial SFT; access to a reward function.	High (requires training a policy model with RL, which is computationally intensive).	Building an adaptive, real-time prompt optimization system, especially for non-text modalities.

Evaluation Protocols and Performance Benchmarks

A robust evaluation framework is critical for quantifying the improvement provided by an

optimized prompt. The PromptWizard repository and the broader field of prompt engineering employ several methods to measure performance, ranging from simple, objective metrics to more complex, model-driven assessments.

Measuring Prompt Quality Improvement

The method for evaluating a prompt's effectiveness is highly dependent on the nature of the task.

- **For Quantitative Tasks:** For problems that have a single, verifiable correct answer, such as mathematics or multiple-choice questions, PromptWizard employs a direct answer matching strategy.[1] The framework is designed to work with a user-defined Python function,
  extract_final_answer(), which is tailored to parse the LLM's output. This function relies on the prompt's answer_format instruction to ensure the output is structured predictably (e.g., extracting the numerical value that follows a #### token).[1] The extracted answer is then compared against the ground-truth value from the dataset to determine correctness.
- **For Qualitative Tasks:** For tasks involving open-ended generation, such as summarization, creative writing, or complex explanations, a simple string match is insufficient. In these cases, the PromptWizard documentation recommends adopting the **"LLM-as-a-Judge"** approach.[1] This paradigm uses a powerful, state-of-the-art LLM (often GPT-4) as an impartial evaluator to score or rank the outputs of other models.[13] The judge LLM is given the initial query, the generated response, and a detailed rubric or set of evaluation criteria (e.g., "Assess the following summary for coherence, accuracy, and conciseness on a scale of 1 to 10"). This method has become a standard practice for evaluating nuanced, qualitative tasks where human evaluation is costly and traditional metrics like ROUGE or BLEU may not fully capture quality.[15]

Benchmark Datasets Referenced

The PromptWizard repository explicitly mentions several datasets used for its development and evaluation, primarily focusing on mathematical and logical reasoning.

- **GSM8K (Grade School Math 8K):** This is the primary example dataset used throughout the PromptWizard documentation.[1] It contains approximately 8,500 high-quality, linguistically diverse grade school math word problems that require 2-8 steps of reasoning to solve.[5] Its structure, which includes detailed step-by-step solutions, makes it an ideal benchmark for evaluating CoT capabilities.[12]
- **SVAMP and AQUA-RAT:** These datasets are also mentioned as being suitable for the PromptWizard evaluation framework.[1] Like GSM8K, they are collections of math word problems with numerical answers, which allows for straightforward, objective evaluation

via direct answer matching.

- **Instruction Induction (BBII):** The performance profile curves presented in the research associated with PromptWizard use the Big Bench Instruction Induction (BBII) task suite for evaluation.[1] This benchmark tests a model's ability to infer the underlying task from a few examples, a core aspect of in-context learning that PromptWizard aims to optimize.

Broader Reasoning Benchmarks (MATH and HumanEval)

While not directly integrated into the PromptWizard codebase, the MATH and HumanEval benchmarks are essential for a comprehensive evaluation of any model intended for complex reasoning and instruction-following tasks.

- **MATH Dataset:** This dataset, curated by Hendrycks et al., represents a significant step up in difficulty from GSM8K. It consists of 12,500 problems from high school math competitions, covering subjects like algebra, geometry, number theory, and calculus.[17] It serves as a standard benchmark for assessing the advanced mathematical reasoning capabilities of frontier LLMs.[18]
- **HumanEval:** Developed by OpenAI, HumanEval is the industry-standard benchmark for evaluating an LLM's ability to generate functionally correct code.[20] It comprises 164 hand-written programming problems, each with a function signature, a docstring explaining the task, and a set of unit tests.[20] Performance is measured using the pass@k metric, which calculates the probability that at least one of k generated code samples passes all unit tests.[21]

For the goal of developing a commercial-grade prompt enhancement model, evaluating the final fine-tuned Qwen3-30B on these more challenging benchmarks is critical. Success on GSM8K demonstrates foundational reasoning, but performance on MATH and HumanEval would provide stronger evidence of the model's general-purpose problem-solving and instruction-following prowess, ensuring that the specialization for prompt enhancement has not degraded its core capabilities.

Curated Catalog of Open-Source Datasets for Prompt Enhancement

The selection of appropriate training data is the most critical factor in the success of fine-tuning a model for prompt enhancement. The primary challenge for a commercial application is not just finding relevant data, but finding data with a permissive license that allows for its use in a proprietary, for-profit product. A significant portion of well-known instruction-tuning datasets, such as the original Stanford Alpaca, are released under non-commercial licenses (e.g., CC BY-NC 4.0), rendering them legally unusable for this

project's stated goal.[22]
The following catalog, therefore, prioritizes datasets with clear, commercially permissive licenses (e.g., Apache 2.0, MIT) or provides actionable strategies for generating such datasets. Each entry includes details on its structure, size, license, and direct relevance to the task of transforming vague user inputs into well-engineered prompts.

Master Dataset Compendium for Prompt Enhancement

Dataset Name	Direct Link	Primary Use Case	Data Format	Size	Commercial Use License	Example / Structural Notes
Prompt Engineering Dataset	Kaggle	Prompt Transformation	CSV	1,000 examples	Unspecified (High Risk)	Direct weak/vague prompt -> effective prompt pairs. Ideal structure but license must be verified.[24]
gokaygokay/prompt-enhancer-dataset	Hugging Face	Prompt Transformation	Parquet	~18k examples	Apache 2.0 (Inferred)	The model trained on this data is Apache 2.0. Consists of high-quality, descriptive prompts. Good for learning the target style.[25]
theminji/midjourney-prompt-enhancement	Hugging Face	Prompt Transformation	CSV	490 examples	Unspecified (High Risk)	input (simple) -> output (enhanced) pairs for image generation.

						Demonstrates context/style expansion. [27]
H2O.ai WizardLM Implementation	GitHub	Instruction Following (Generation)	Python script generates JSON	User-defined	**Apache 2.0**	Provides code to replicate the Evol-Instruct method, creating a commercially safe, high-complexity instruction dataset. [10]
ShareGPT (Permissive Variants)	Hugging Face	Instruction Following	JSON / Parquet	Varies (e.g., 10k-100k)	Varies (e.g., Apache 2.0)	User must filter for commercially licensed versions. Provides rich, multi-turn conversational data. [28]
GSM8K	Hugging Face	Chain-of-Thought Reasoning	JSON Lines (.jsonl)	~8.5k examples	**MIT License**	question and answer pairs where the answer includes step-by-step reasoning. Excellent for teaching CoT. [5]
abacusai/Long-Context Datasets	GitHub	Context Expansion	JSON	Varies	Unspecified (High Risk)	Provides datasets (e.g., WikiQA) and methods (RAFT) for

						creating long-context training data with distractor documents.[30]

In-Depth Analysis of High-Priority Datasets

5.1. Prompt Transformation Datasets (Simple → Enhanced)

These datasets are the most valuable as they provide direct supervised examples of the target task.

- **gokaygokay/prompt-enhancer-dataset**: This dataset is a strong candidate for commercial use. While the dataset card itself does not specify a license, the model card for gokaygokay/Flux-Prompt-Enhance, which explicitly states it was trained on this dataset, is licensed under the permissive **Apache 2.0** license.[25] This provides a strong indication that the data is also intended for open use. The data consists of highly descriptive, well-structured prompts, making it an excellent resource for teaching a model the characteristics of a high-quality output.[25]
- **theminji/midjourney-prompt-enhancement**: This dataset provides perfect examples of prompt transformation, showing how a simple idea is expanded with stylistic details, technical parameters, and descriptive keywords.[27] For example, "A bustling city street scene at night" is transformed into "Photorealistic bustling city street at night, vibrant lights, busy pedestrians, urban life --ar 16:9 --v 5.2".[27] However, its utility is severely hampered by the lack of an explicit license. Under standard copyright law, this data cannot be used for a commercial product without explicit permission from the author.[32]
- **Kaggle "Prompt Engineering Dataset"**: Described as containing 1,000 examples of transformations from weak to effective prompts, this dataset appears to be ideally structured for the user's fine-tuning task.[24] It covers 16 different categories and demonstrates 11 distinct prompting techniques. Unfortunately, the resource was inaccessible during analysis, and Kaggle datasets carry their own specific licenses which must be carefully reviewed.[24] Securing access to and verifying the license of this dataset should be a high priority.

5.2. Instruction-Following Datasets (Commercially Permissive)

These datasets are essential for building the model's foundational ability to follow complex instructions before specializing it.

- **WizardLM / Evol-Instruct Method**: The most significant challenge with many instruction datasets is their restrictive licensing. The Evol-Instruct methodology provides a powerful, strategic solution. Instead of using a pre-existing, restrictively licensed dataset, one can use the methodology itself to generate a new, proprietary, or permissively licensed dataset. The open-source, **Apache 2.0** licensed implementation from H2O.ai provides the direct tooling to do so.[10] This involves taking a small set of seed prompts and using a powerful LLM (like GPT-4 via API) to iteratively rewrite and complexify them. This creates a large-scale, high-quality dataset that is commercially safe to use for fine-tuning. This is a key recommended path.

5.3. Specialized Datasets (Context Expansion & Task Decomposition)

For more advanced prompt enhancement capabilities, such as adding necessary context or breaking down complex requests, specialized data is required.

- **Context Expansion**: The concept of **Retrieval Augmented Fine-Tuning (RAFT)** offers a robust method for synthetically generating training data that teaches a model to handle long contexts.[35] The process involves taking a standard question-answer pair and injecting several "distractor" documents into the context alongside the single document containing the correct answer. This forces the model to learn to identify and utilize the relevant information within a noisy, extended context. The abacusai/Long-Context repository provides datasets created with this technique, such as WikiQA Altered_Numeric_QA, which modifies answers to prevent the model from relying on its pre-trained knowledge.[30]
- **Task Decomposition**: While no off-the-shelf dataset for "simple prompt -> decomposed prompt" was identified, research points to foundational datasets that can be used to synthesize this data. The **AsyncHow** dataset, used for evaluating agentic workflows, contains complex tasks represented as sequential and parallel task graphs.[36] One could use this structured data to generate training pairs where a high-level goal (e.g., the name of the task graph) is the input, and the structured, step-by-step plan is the desired output prompt. This would teach the model to transform a vague request like "Plan a product launch" into a detailed, actionable sequence of steps.

Strategic Recommendations for Training Qwen3-30B

Based on the comprehensive analysis of the PromptWizard framework, related Microsoft research, and the available dataset landscape, the following strategic blueprint is recommended for training the Qwen3-30B model to serve as a real-time prompt enhancement engine for a commercial SaaS product.

A Data-Centric, Multi-Stage Training Blueprint

A multi-stage fine-tuning approach is recommended to systematically build the required capabilities into the Qwen3-30B model. This approach ensures that foundational skills are established before introducing more specialized tasks, leading to a more robust and capable final model.

- **Stage 1: Foundational Instruction Tuning.** The primary goal of this stage is to enhance the base Qwen3-30B model's general ability to understand and follow complex, nuanced instructions. This is a prerequisite for the more specific task of prompt enhancement.
    - **Recommended Data:** A large, diverse, and commercially permissive instruction-following dataset should be created or curated. The most viable strategy is to use the **Evol-Instruct methodology**, leveraging the Apache 2.0 licensed H2O.ai implementation.[10] This involves generating a dataset of at least 70,000-100,000 complex instruction-response pairs. This can be blended with a commercially-cleared variant of the ShareGPT dataset to increase conversational diversity.
    - **Objective:** To produce a model with strong general-purpose instruction-following capabilities, which will serve as the base for the next stage.
- **Stage 2: Specialization on Prompt Transformation.** With a strong instruction-following foundation, the model is now ready to be specialized for the core task of transforming vague inputs into well-engineered prompts.
    - **Recommended Data:** This stage requires a smaller, highly focused dataset of "simple prompt" → "enhanced prompt" pairs. The gokaygokay/prompt-enhancer-dataset provides an excellent source for the target "enhanced prompt" style.[25] This can be combined with prompts generated from the Kaggle Prompt Engineering Dataset (pending license verification).[24] If necessary, a synthetic dataset can be created by using GPT-4 to generate "enhanced" versions of simple prompts drawn from other datasets.
    - **Objective:** To fine-tune the model from Stage 1 specifically on the task of prompt refinement, teaching it to add clarity, context, constraints, and structure to user inputs.
- **Stage 3 (Optional): Reasoning Enhancement.** If the desired "enhanced" prompts frequently require the model to generate structured reasoning or step-by-step plans (i.e., Chain-of-Thought), a final, brief fine-tuning stage can be beneficial.
    - **Recommended Data:** The **GSM8K** dataset, with its MIT license, is ideal for this purpose.[12] Fine-tuning on its question-and-reasoning-answer pairs will reinforce the model's ability to generate and structure CoT-style outputs.
    - **Objective:** To improve the model's capacity for generating prompts that include explicit, auditable reasoning steps, which can be a valuable feature for complex

tasks.

Dataset Integration and Formatting Plan

A unified data format is crucial for an efficient training pipeline. It is recommended to preprocess and convert all selected datasets into a consistent JSON Lines (.jsonl) format, with each line containing a dictionary with fields such as instruction, input, and output. For the prompt transformation task in Stage 2, this would map to:

- instruction: A static instruction for the model, e.g., "Rewrite the following user query to be a detailed, effective prompt for a large language model."
- input: The original, vague user prompt.
- output: The target, well-engineered prompt.

Data quality is paramount. Aggressive cleaning and curation should be performed to remove low-quality, irrelevant, or malformed examples from all datasets before they are used for fine-tuning.

Evaluation Framework for Commercial Deployment

A multi-faceted evaluation strategy is essential to ensure the model is not only effective but also reliable and safe for deployment in a commercial product.

1. **Offline Evaluation:** Before any deployment, the fine-tuned model should be rigorously tested on a held-out set of prompt transformation pairs. Performance can be measured using a combination of lexical and semantic metrics. N-gram-based metrics like ROUGE and BLEU can provide a baseline, while more advanced, embedding-based metrics like BERTScore will offer a more accurate measure of semantic similarity between the model's generated prompt and the ground-truth enhanced prompt.
2. **LLM-as-a-Judge:** For a more nuanced assessment of quality, a powerful proprietary model like GPT-4o should be employed as an automated judge. The judge can be prompted with a detailed rubric to score the enhanced prompts on dimensions such as **Clarity**, **Specificity**, **Inclusion of Necessary Context**, and **Absence of Ambiguity**. Comparing the scores for prompts generated by the fine-tuned Qwen3 model against the ground-truth prompts provides a scalable measure of quality.
3. **Human Evaluation:** As the gold standard, blind A/B testing with human evaluators is non-negotiable for a commercial product. In this setup, end-users or internal testers interact with two versions of the system: one that uses the user's raw, vague prompt, and one that uses the prompt enhanced by the fine-tuned Qwen3 model. The final outputs from the downstream LLM are then presented to human raters who judge their quality, relevance, and helpfulness without knowing which prompt was used. A statistically significant preference for the outputs from the enhanced prompts is the ultimate validation of the model's effectiveness.
4. **Regression Testing on Benchmarks:** After the specialization fine-tuning in Stage 2

and 3, it is critical to evaluate the model on general reasoning benchmarks like **GSM8K** and **MATH**.[5] This step ensures that the specialization process has not led to "catastrophic forgetting," where the model loses its general problem-solving abilities. Maintaining a stable or only slightly degraded baseline performance on these benchmarks is a key indicator of a successful and robust fine-tuning process.

Works cited

1. microsoft/PromptWizard: Task-Aware Agent-driven Prompt ... - GitHub, accessed August 7, 2025, https://github.com/microsoft/PromptWizard
2. PromptWizard Task-Aware Prompt Optimization Framework - Microsoft Open Source, accessed August 7, 2025, https://microsoft.github.io/PromptWizard/
3. Microsoft AI Research Open-Sources PromptWizard: A Feedback-Driven AI Framework for Efficient and Scalable LLM Prompt Optimization : r/machinelearningnews - Reddit, accessed August 7, 2025, https://www.reddit.com/r/machinelearningnews/comments/1hhf447/microsoft_ai_research_opensources_promptwizard_a/
4. Chain-of-Thought (CoT) Prompting - Prompt Engineering Guide, accessed August 7, 2025, https://www.promptingguide.ai/techniques/cot
5. README.md · openai/gsm8k at main - Hugging Face, accessed August 7, 2025, https://huggingface.co/datasets/openai/gsm8k/blob/main/README.md
6. microsoft/LMOps: General technology for enabling AI ... - GitHub, accessed August 7, 2025, https://github.com/microsoft/LMOps
7. Optimizing Prompts for Text-to-Image Generation - arXiv, accessed August 7, 2025, https://arxiv.org/html/2212.09611v2
8. [2212.09611] Optimizing Prompts for Text-to-Image Generation - arXiv, accessed August 7, 2025, https://arxiv.org/abs/2212.09611
9. LLMs build upon Evol Insturct: WizardLM, WizardCoder, WizardMath - GitHub, accessed August 7, 2025, https://github.com/nlpxucan/WizardLM
10. h2oai/h2o-wizardlm: Open-Source Implementation of WizardLM to turn documents into Q:A pairs for LLM fine-tuning - GitHub, accessed August 7, 2025, https://github.com/h2oai/h2o-wizardlm
11. Automatic Instruction Evolving for Large Language Models - ACL Anthology, accessed August 7, 2025, https://aclanthology.org/2024.emnlp-main.397.pdf
12. openai/grade-school-math - GitHub, accessed August 7, 2025, https://github.com/openai/grade-school-math
13. Evaluating Scoring Bias in LLM-as-a-Judge - arXiv, accessed August 7, 2025, https://arxiv.org/html/2506.22316v1
14. LLMs-as-Judges: A Comprehensive Survey on LLM-based Evaluation Methods - arXiv, accessed August 7, 2025, https://arxiv.org/html/2412.05579v2
15. Learning to Plan & Reason for Evaluation with Thinking-LLM-as-a-Judge - arXiv, accessed August 7, 2025, https://arxiv.org/pdf/2501.18099
16. A Survey on LLM-as-a-Judge - arXiv, accessed August 7, 2025, https://arxiv.org/html/2411.15594v4
17. hendrycks/math: The MATH Dataset (NeurIPS 2021) - GitHub, accessed August 7,

2025, https://github.com/hendrycks/math
18. MATH Benchmark (Mathematics Assessment of Textual Heuristics) - Klu.ai, accessed August 7, 2025, https://klu.ai/glossary/math-eval
19. Meta Prompting | Prompt Engineering Guide, accessed August 7, 2025, https://www.promptingguide.ai/techniques/meta-prompting
20. HumanEval Benchmark - Klu.ai, accessed August 7, 2025, https://klu.ai/glossary/humaneval-benchmark
21. What is HumanEval ? | Deepchecks, accessed August 7, 2025, https://www.deepchecks.com/glossary/humaneval/
22. Ecosystem Graphs for Foundation Models - Stanford CRFM, accessed August 7, 2025, https://crfm.stanford.edu/ecosystem-graphs/index.html?asset=Alpaca%20dataset
23. tatsu-lab/stanford_alpaca: Code and documentation to train Stanford's Alpaca models, and generate the data. - GitHub, accessed August 7, 2025, https://github.com/tatsu-lab/stanford_alpaca
24. Prompt Engineering Dataset | Kaggle, accessed August 7, 2025, https://www.kaggle.com/datasets/austinfairbanks/prompt-engineering-dataset
25. gokaygokay/prompt-enhancer-dataset - Hugging Face, accessed August 7, 2025, https://huggingface.co/datasets/gokaygokay/prompt-enhancer-dataset
26. gokaygokay/Flux-Prompt-Enhance at 5b465f639a77b129b0e876120ee04ab6a78c724a - Hugging Face, accessed August 7, 2025, https://huggingface.co/gokaygokay/Flux-Prompt-Enhance/tree/5b465f639a77b129b0e876120ee04ab6a78c724a
27. theminji/midjourney-prompt-enhancement · Datasets at Hugging Face, accessed August 7, 2025, https://huggingface.co/datasets/theminji/midjourney-prompt-enhancement
28. ShareGPT Datasets - a bunnycore Collection - Hugging Face, accessed August 7, 2025, https://huggingface.co/collections/bunnycore/sharegpt-datasets-66fa831dcee14c587f1e6d1c
29. Chat Datasets — torchtune 0.3 documentation, accessed August 7, 2025, https://docs.pytorch.org/torchtune/0.3/basics/chat_datasets.html
30. abacusai/Long-Context: This repository contains code and tooling for the Abacus.AI LLM Context Expansion project. Also included are evaluation scripts and benchmark tasks that evaluate a model's information retrieval capabilities with context expansion. We also include key experimental results and instructions for reproducing and building on them. - GitHub, accessed August 7, 2025, https://github.com/abacusai/Long-Context
31. gokaygokay/Flux-Prompt-Enhance - Hugging Face, accessed August 7, 2025, https://huggingface.co/gokaygokay/Flux-Prompt-Enhance
32. Licenses - Hugging Face, accessed August 7, 2025, https://huggingface.co/docs/hub/repositories-licenses
33. Dataset Cards - Hugging Face, accessed August 7, 2025,

https://huggingface.co/docs/hub/datasets-cards
34. accessed December 31, 1969, httpss://www.kaggle.com/datasets/austinfairbanks/prompt-engineering-dataset
35. Extending LLM context with 99% less training tokens - Cerebras, accessed August 7, 2025, https://www.cerebras.ai/blog/extending-llm-context-with-99-less-training-tokens
36. Advancing Agentic Systems: Dynamic Task Decomposition, Tool Integration and Evaluation using Novel Metrics and Dataset - arXiv, accessed August 7, 2025, https://arxiv.org/html/2410.22457v1