Project Helios: Automating Predictive Signal Extraction

Alex Pan
Founder, Edictive

Abstract

Large Language Models (LLMs) can improve time-series forecasting by incorporating textual context (Williams et al., 2025), but the antecedent task of extracting predictive signals from unstructured text remains a critical bottleneck. We introduce Project Helios, a conceptual framework that operationalizes the hypothesis that this process can be automated by unifying synthetic data generation and automatic prompt optimization (APO). Helios's two-pillar architecture features (i) *Auto-Foundry*, a minimal-configuration data generator using correlated sampling (Kowshik et al., 2024) and hard-case mining (Wu et al., 2024), and (ii) *Helios-Ensemble*, a multi-objective prompt optimizer leveraging a novel hybrid algorithm, which we term ParetoStraGo 2.0, that synthesizes concepts from Pareto-front optimization (Zhao et al., 2025) and strategic guidance (Wu et al., 2024). By tightly coupling these components, Helios provides a turnkey solution that transforms a high-level signal definition into an optimized, production-ready prompt, significantly accelerating the experimental cycle for signal engineering. This paper's contributions are: (1) the conceptual architecture, (2) a map of key design decisions, (3) a preliminary feasibility analysis, and (4) a research agenda, all motivated by the scaling imperatives of modern AI (Kaplan et al., 2020).

1 Introduction

The application of Large Language Models (LLMs) to time-series analysis is an increasingly prominent field, driven by their unique ability to integrate rich, unstructured textual context into the forecasting process (Williams et al., 2025; Ansari et al., 2024; Jin et al., 2024). This capacity for contextual understanding allows LLMs to outperform traditional models in scenarios where external information, conveyed through natural language, is critical for accurate prediction. A central challenge in enterprise forecasting is the extraction of these predictive signals from vast quantities of unstructured text, such as sales call transcripts, support tickets, and operational logs, which often contain high-value leading indicators of future outcomes.

However, the process of transforming this raw text into structured, predictive features is a significant bottleneck. The typical workflow is slow, expensive, and bespoke. It involves human annotators labeling thousands of examples, a process with significant operational overhead, followed by ML engineers fine-tuning specialized models to extract the desired signals. This multi-stage, human-in-the-loop cycle not only is costly but also severely limits the speed of experimentation, making it impractical to test a wide combinatorial space of potential signals.

We hypothesize that the same contextual understanding that LLMs leverage for forecasting can be repurposed to address this fundamental challenge of signal extraction. The process of identifying a signal, such as 'consequence awareness' in a sales transcript, is itself a predictive task conditioned on a user-provided rubric. An LLM's ability to generalize from instructions makes it adept at this form of rubric-guided labeling (Shin et al., 2020). To operationalize this insight, we turn to two rapidly advancing fields: synthetic data generation (Patel et al., 2024; Kowshik et al., 2024) and automatic prompt optimization (APO) (Yang et al., 2024; Zhao et al., 2025).

While prior work has explored these fields separately, a significant gap exists in their integration. Existing systems do not use the downstream optimization task to guide the data generation process, creating a disconnect where a data generator operates blind to the needs of the optimizer LLM. Project Helios is designed to fill this gap by creating a tightly-coupled, self-improving system where the data generator is explicitly tasked with producing data that is maximally informative for discovering an optimal, multi-objective prompt.

This paper makes the following primary contributions:

- We introduce **Project Helios**, a novel conceptual framework that automates predictive signal extraction by tightly coupling synthetic data generation with multi-objective prompt optimization.
- We present **Auto-Foundry**, a synthetic data generation pillar that uses correlated sampling and hard-case mining to produce diverse and challenging datasets tailored to a specified signal rubric.
- We detail **Helios-Ensemble**, a multi-objective prompt optimization pillar featuring a novel hybrid algorithm, ParetoStraGo 2.0, which efficiently discovers prompts that are Pareto-optimal across multiple signal-extraction objectives.

This paper is structured as follows. Section 2 traces the evolution of signal-extraction solutions. Section 3 formalizes the problem and key design decisions. Section 4 presents the high-level Helios architecture, followed by detailed descriptions of its two pillars in Sections 5 and 6. We discuss feasibility in Section 7, present related work in Section 9, and conclude in Section 10.

2 Evolution of Signal-Extraction Solutions

This section traces the evolution of solutions for extracting predictive signals from unstructured text, framing it in three stages. We begin with the traditional, manual-intensive paradigm and its inherent limitations. We then examine the current stage of partial automation, where synthetic data generation and prompt optimization are treated as separate, disconnected processes. Finally, we articulate the vision for Project Helios as the third stage: a tightly-coupled, fully-automated architecture that represents the logical and necessary next step in scaling signal engineering for enterprise applications.

2.1 Stage 1: Human Annotation and Bespoke Models

The conventional approach to signal extraction is a direct but costly process rooted in manual human effort. In this paradigm, domain experts or annotators are tasked with meticulously labeling thousands of text examples according to a predefined rubric. For instance, to create a signal for "customer churn risk," annotators would read through support tickets and manually flag instances exhibiting frustration or explicit threats to cancel a service. This labeled dataset is then used by machine learning engineers to train or fine-tune a bespoke model, often a specialized classifier, to automate the signal extraction on new, unseen data.

This stage is characterized by significant drawbacks in time, cost, and scalability. A more subtle but equally critical failure mode arises from the nature of the data itself. Datasets created via manual annotation of a raw corpus often reflect the natural, imbalanced distribution of events, where high-value signals are rare occurrences (Yu et al., 2023). Models trained on such data become biased towards the majority class, leading to poor performance on the underrepresented minority classes. This deficit is directly captured by a low macro F1 score, which penalizes poor performance on rare classes. This inflexibility and performance skew make it impractical for organizations to explore a wide range of potential predictive signals, severely limiting the speed of hypothesis testing and model development.

2.2 Stage 2: Partial Automation with Decoupled Components

The limitations of manual annotation have driven the development of partially automated solutions. These solutions typically follow one of two independent paths: generating synthetic data to replace human annotation or using automated prompt optimization (APO) to refine LLM instructions. The first path focuses exclusively on **synthetic data generation**. Techniques like back-translation

(Sennrich et al., 2016) or direct generation from LLMs using tools like DataDreamer (Patel et al., 2024) can create large volumes of labeled data at a fraction of the cost of manual annotation. However, when decoupled from the downstream optimization task, the generator operates in a vacuum. It may produce a dataset that lacks the diversity or challenging examples needed to discover a robust, generalizable prompt, leading to "easy" data that results in over-fitted solutions. The second path concentrates on **automatic prompt optimization**. This field has explored methods ranging from discrete textual edits based on pseudo-gradients (Pryzant et al., 2023) to the tuning of continuous soft prompts (Lester et al., 2021). While powerful, APO methods are constrained by the quality of the evaluation dataset. If the data lacks diversity or fails to represent hard-to-classify edge cases, the optimizer LLM may converge on a prompt that performs well on the limited evaluation set but fails to generalize to real-world data. Furthermore, isolated APO systems often struggle to balance the conflicting, multi-objective nature of signal extraction (e.g., accuracy vs. latency vs. cost).

2.3 Stage 3: Tightly-Coupled Generation and Optimization

The core insight of Project Helios is that the independent paths of Stage 2 are destined to merge. A truly automated and effective system requires tight coupling between the data generator and the prompt optimizer, creating a closed-loop system where each component informs and improves the other. The limitations of decoupled systems are clear: generators risk producing monotonous data that lacks the diversity and challenging edge cases required for robust optimization, a challenge of bias and mode collapse well-documented in synthetic data literature (Yu et al., 2023). Simultaneously, optimizers risk "vacuum optimization," perfecting a prompt for a non-representative dataset. The vision for Helios is a Stage 3 architecture that unifies these processes. In this model, the prompt optimizer's performance on its multi-objective function provides direct feedback to the data generator. The generator, in turn, is tasked with producing data that is maximally informative for the optimizer LLM. This includes leveraging techniques like correlated sampling to ensure data diversity (Kowshik et al., 2024) and hard-case mining to surface examples where the current optimal prompt is weakest. This integrated approach is not merely an improvement but a necessity, driven by the scaling imperatives of modern AI (Kaplan et al., 2020), where the sheer volume of data and complexity of signals demand an end-to-end, zero-touch automation framework.

3 Problem Formulation & Key Design Decisions

The central objective is to create a system that transforms a user-defined rubric and a corpus of raw text into a set of reliable, multi-metric predictive signals, embodied by one or more optimized prompts. This requires navigating a series of critical trade-offs:

- **D1:** Real vs. Synthetic Data Blend
- **D2:** Diversity vs. Rubric Fidelity
- **D3:** Single- vs. Multi-Objective Optimization
- **D4:** Generator–Judge Independence
- **D5:** Optimizer-Guided Data Enrichment
- **D6:** Architectural Modularity & Evolvability

For each of the following six design decisions, we articulate the competing forces, justify the choice made in the Helios architecture, and state the anticipated benefit.

This section formalizes the core problem that Project Helios is designed to solve and outlines the key architectural and algorithmic design decisions that shape our proposed solution. The central objective is to create a system that transforms a user-defined rubric and a corpus of raw text into a set of reliable, multi-metric predictive signals, embodied by one or more optimized prompts. Achieving this requires navigating a series of critical trade-offs, from the composition of the training data to the nature of the optimization process itself. For each of the following six design decisions, we articulate the competing forces, justify the choice made in the Helios architecture, and state the anticipated benefit.

D1: Real vs. Synthetic Data Blend

Forces: The choice of data source presents a fundamental trade-off between fidelity and scalability. Real-world, human-annotated data provides the highest-fidelity ground truth but is prohibitively expensive and slow to acquire at scale. Conversely, synthetic data generated by LLMs is cheap and scalable but risks being unrealistic, lacking diversity, or drifting from the target distribution (Patel et al., 2024).

Helios Choice: Helios adopts a hybrid, synthetic-first approach. While the system can be bootstrapped with a small seed of real data, its primary operational mode relies on the *Auto-Foundry* pillar to generate a large, diverse corpus of synthetic data tailored to the user's rubric.

Anticipated Benefit: This choice makes signal exploration economically feasible. By minimizing reliance on manual annotation, Helios allows teams to test a wide array of potential signals without incurring significant operational costs or delays, democratizing the signal engineering process.

D2: Diversity vs. Rubric Fidelity

Forces: A synthetic data generator must balance two competing objectives: rubric fidelity and data diversity. High fidelity ensures that the generated examples are relevant to the specified signal. However, an overemphasis on fidelity can lead to mode collapse, where the generator produces monotonous and un-diverse examples that fail to cover the full semantic space of the problem (Yu et al., 2023). A diverse dataset, conversely, is essential for training a robust and generalizable prompt optimizer.

Helios Choice: Helios manages this trade-off by structurally separating the concerns of diversity and fidelity. The *Auto-Foundry* pillar uses techniques like correlated sampling (Kowshik et al., 2024) to generate a diverse set of candidate examples, while a separate, independent Judge LLM acts as a quality gate to enforce rubric fidelity.

Anticipated Benefit: This decoupling produces a dataset that is both highly relevant to the user's task and sufficiently diverse to train a robust, generalizable prompt optimizer that avoids overfitting to narrow data distributions.

D3: Single- vs. Multi-Objective Optimization

Forces: Signal extraction is inherently a multi-objective problem. While accuracy is a primary goal, real-world applications must also consider trade-offs with inference latency, computational cost, and the explainability of the extracted signal. Single-objective optimization is simpler to implement but fails to capture these critical operational constraints.

Helios Choice: The *Helios-Ensemble* pillar explicitly embraces multi-objective optimization, building on a Pareto optimization framework (Zhao et al., 2025). It aims to discover not a single "best" prompt, but a set of Pareto-optimal prompts that represent the best possible trade-offs across user-defined objectives.

Anticipated Benefit: This approach delivers a menu of production-ready prompts, empowering users to select the optimal solution for their specific operational context (e.g., choosing a slightly less accurate but much faster prompt for a real-time application).

D4: Generator–Judge Independence

Forces: Within the synthetic data pipeline, using the same LLM to both generate examples and judge their quality is computationally cheaper. However, this creates a significant risk of self-reinforcement bias, where a model is overly lenient on its own outputs or those from a model with a similar architecture. Using a sequence-to-sequence model like BART (Lewis et al., 2019) for generation is an alternative, but it may lack the zero-shot power of a large, frozen foundation model.

Helios Choice: Helios enforces functional independence between the Generator and Judge LLMs within the *Auto-Foundry* pillar. These roles are fulfilled by distinct models (e.g., from different providers) or, at a minimum, separate API endpoints with distinct system prompts and configurations to prevent information leakage and self-reinforcement bias.

Anticipated Benefit: This separation mitigates evaluation bias and ensures a more objective and reliable quality gate for the synthetic data. This leads to a higher-quality training corpus for the prompt optimizer, which is critical for discovering robust and effective prompts.

D5: Optimizer-Guided Data Enrichment

Forces: To accelerate learning and improve robustness, it is beneficial to focus the data generation process on "hard cases"—examples where the current best prompt fails or performs poorly. However, an excessive focus on these failure modes can lead to overfitting on noise or outliers in the data.

Helios Choice: Helios implements an iterative, closed-loop enrichment process. The *Helios-Ensemble* optimizer identifies its weakest-performing areas, and this feedback is passed to the *Auto-Foundry*. This mechanism is inspired by iterative refinement loops where model failures guide subsequent improvements. For instance, some prompt optimizers generate textual "gradients" from error cases to guide the next optimization step (Pryzant et al., 2023), while others use strategic analysis of both successes and failures to guide refinement (Wu et al., 2024). Auto-Foundry is explicitly tasked with generating new synthetic data that targets these identified weaknesses.

Anticipated Benefit: This creates an adaptive training curriculum of increasing difficulty. It forces the prompt optimizer to continuously improve its generalization capabilities by confronting its own failure modes, leading to more robust and reliable signal extraction.

D6: Architectural Modularity & Future Evolution

Forces: The foundation model landscape is evolving at an extremely rapid pace. A monolithic, tightly-integrated system architecture risks becoming brittle and quickly obsolete. A modular, micro-kernel-like architecture (Bass et al., 2003), while potentially more complex to design initially, offers greater adaptability and long-term viability.

Helios Choice: Helios is designed as a modular, two-pillar system with well-defined interfaces. The architecture explicitly anticipates that many of its internal components (e.g., the generator LLM, the judge LLM) will eventually be absorbed or replaced by more powerful, general-purpose foundation models. The system is designed to allow for this evolution gracefully.

Anticipated Benefit: This micro-kernel-inspired design ensures the long-term relevance and evolvability of the Helios framework. Core components can be swapped, upgraded, or deprecated without necessitating a complete system redesign, allowing the architecture to adapt as the foundation model ecosystem matures.

4 Project Helios Architecture (High-Level)

This section presents the high-level architecture of Project Helios, a framework designed for modularity, evolvability, and responsible AI by design. To structure our discussion, we adopt the three-layer architectural view proposed by (Lu et al., 2024): a *System Layer* containing the deployed components, an *Operation Layer* providing the core AI tooling and logic, and a *Supply-Chain Layer* that sources the underlying foundation models. We first present a conceptual diagram of the architecture and then walk through the data flow, illustrating how the system transforms a user's rubric into a production-ready, optimized prompt. We conclude by discussing the key decoupling points that ensure the system's long-term viability.

The end-to-end workflow of Project Helios, depicted in Figure 1, begins with the user providing two primary inputs to the System Layer: a natural-language **rubric** defining the predictive signal and an optional corpus of raw **text**. These inputs initiate a process within the Operation Layer, starting with the **Auto-Foundry** pillar. Auto-Foundry's mission is to generate a high-quality synthetic dataset tailored to the rubric. It leverages a Generator LLM, sourced from the Supply-Chain Layer, to create diverse candidate examples and a separate Judge LLM to ensure rubric fidelity (satisfying D1, D2, D4). This dataset then serves as the training and evaluation corpus for the **Helios-Ensemble** pillar. The Helios-Ensemble optimizer uses this data to discover a set of Pareto-optimal prompts that balance user-defined objectives like accuracy, latency, and cost (D3). Crucially, the system forms a closed loop: performance metrics from the optimizer LLM, particularly on hard-to-classify examples, are fed back to Auto-Foundry. This feedback loop enables targeted hard-case mining, instructing the data

Figure 1: The three-layer architecture of Project Helios, adapted from (Lu et al., 2024). This layered view separates concerns: the Supply-Chain provides the base models, the Operation Layer contains the core logic of Helios, and the System Layer handles user interaction. This modularity is key to the framework's evolvability.

generator to produce new examples that address the optimizer LLM's current weaknesses (D5). The cycle of generation, optimization, and feedback continues until a stable, high-performance prompt is produced and delivered to the user.

A central design philosophy of Helios is architectural modularity and evolvability, inspired by the micro-kernel pattern (Bass et al., 2003). We anticipate that the capabilities of foundation models will continue to expand, eventually absorbing functions currently handled by distinct components within our architecture. For example, a future, more powerful LLM might possess innate multi-objective optimization capabilities, rendering a separate optimizer module redundant. The Helios architecture is designed to accommodate this evolution by treating its core components—the Generator, Judge, and Optimizer LLMs—as swappable services with well-defined interfaces. This decoupling (D6) ensures that as the foundation models in the Supply-Chain Layer mature, Helios can adapt by simply re-routing internal API calls rather than requiring a fundamental redesign. This approach ensures the framework's long-term viability and relevance in a rapidly changing technological landscape. Table **??** summarizes how the architectural components map to the key design decisions discussed in Section 3.

Architectural Component	**Design Decision(s) Satisfied**
Auto-Foundry (Generator)	D1 (Synthetic-first), D2 (Diversity focus)
Auto-Foundry (Judge)	D2 (Fidelity enforcement), D4 (Independence)
Helios-Ensemble (Optimizer)	D3 (Multi-objective), D5 (Hard-case feedback)
Two-Pillar Modular Structure	D6 (Modularity & Evolvability)

5 Pillar 1 – Auto-Foundry (Zero-Config Synthetic Data)

A Primer on Synthetic Data Generation

Synthetic data generation uses a generative model (often an LLM) to create artificial training examples. The goal is to augment or replace costly human-annotated data. Key challenges include:

- **Mode Collapse:** The generator produces low-diversity, repetitive examples.
- **Distributional Drift:** The synthetic data's statistical properties diverge from real-world data.
- **Bias Amplification:** The generator may amplify societal biases present in its own training data.

Auto-Foundry mitigates these risks through correlated sampling and an independent judge model.

Conceptually, Auto-Foundry acts as an automated data chef, tasked with preparing a varied and high-quality meal (the dataset) according to a customer's recipe (the rubric). For readers primarily focused on prompt optimization, this section details the data-generation engine that creates the high-quality evaluation sets necessary for robust optimization.

The Auto-Foundry pillar serves as the synthetic data engine for Project Helios. Its primary objective is to generate a diverse, high-quality, and rubric-aligned dataset that enables the Helios-Ensemble to discover robust and effective prompts. This process is designed for minimal configuration, requiring

only the user's rubric to initiate a sophisticated data generation pipeline. This section details the key components of Auto-Foundry: its use of correlated sampling for diversity, its closed-loop integration with the optimizer LLM for hard-case mining, its independent quality-gate mechanism, and its modular design for future adaptability.

5.1 Correlated Sampling for Breadth

To prevent the mode collapse common in naive synthetic data generation, Auto-Foundry employs a correlated sampling strategy to ensure dataset diversity (Kowshik et al., 2024).[1]

Instead of generating each data point independently, this technique generates a portfolio of related but distinct examples in parallel. By instructing the Generator LLM to produce examples that cover different semantic facets of the user's rubric (e.g., generating both positive and negative examples of a signal, or examples from different sub-topics), Auto-Foundry ensures the resulting dataset spans a wide area of the problem's semantic space. This breadth is critical for training a prompt optimizer that can generalize beyond a narrow set of examples.

5.2 Optimizer-Guided Data Enrichment

Auto-Foundry operates in a tight feedback loop with the Helios-Ensemble optimizer. This loop facilitates a form of automated curriculum learning, focusing the data generation process on the areas where the current best prompt is weakest. the optimizer LLM identifies misclassified or low-confidence examples—the "hard cases"—and feeds this information back to Auto-Foundry. The Generator LLM is then explicitly tasked with creating new synthetic examples that are semantically similar to these hard cases. This iterative process of identifying and targeting weaknesses, inspired by recent work in prompt optimization that analyzes failure cases (Pryzant et al., 2023; Wu et al., 2024), ensures that the training data continuously challenges the optimizer LLM, pushing it to refine its prompts and improve its generalization capabilities.

5.3 Independent Judge LLM for Quality Gating

To maintain high data quality and enforce rubric fidelity, Auto-Foundry uses an independent Judge LLM as a quality gate. After the Generator LLM produces a batch of diverse synthetic examples, the Judge LLM evaluates each example against the user-provided rubric. Only examples that meet a predefined quality threshold are admitted into the final training corpus. Crucially, the Judge is a separate model instance from the Generator, a design choice that mitigates the risk of self-reinforcement bias where a model might be overly lenient on its own outputs. This separation ensures a more objective and reliable filtering process, leading to a higher-quality dataset for the downstream optimizer.

5.4 Modular Design and Cost Analysis

The Auto-Foundry architecture is designed for modularity, allowing its core components to be swapped to meet different operational needs or to adapt to future technological advances. For instance, the Generator LLM is a pluggable module. A team might choose a smaller model for tasks requiring high fidelity to a specific format, or a larger, more creative foundation model for tasks demanding greater semantic diversity. This flexibility future-proofs the system. From a cost perspective, the primary driver is token consumption by the Generator and Judge LLMs. While generating a large synthetic dataset can be computationally intensive, it is an offline, one-time cost per signal. This initial investment is amortized over the entire lifecycle of the extracted signal, making it significantly more scalable and cost-effective than the continuous operational expense of manual human annotation.

[1] Correlated sampling, as proposed by Kowshik et al. (2024), generates multiple sequences in parallel with strong inter-dependencies. The core hypothesis is that by contrasting the generation of a target example with other examples (either from the same or different classes), the resulting sequences are guided to be more diverse and semantically distinct, mitigating mode collapse.

Algorithm 1: ParetoStraGo 2.0 Optimization Loop (Conceptual)

Input: Initial prompt population P_0, Objective functions O, Strategy set S, Population size k
Output: Pareto-optimal prompt set P^*
$P \leftarrow P_0$;
while *not converged* **do**
  Evaluate each prompt $p \in P$ on objectives O to get scores;
  Update Pareto front P^* with non-dominated prompts from P;
  Select strategy $s \in S$ using bandit algorithm;
  Construct meta-prompt with P^*, scores, and strategy s;
  Generate new candidate prompts P_{new} using Optimizer LLM;
  $P \leftarrow \text{CullToSize}(P \cup P_{new}, k)$;
end

6 Pillar 2 – Helios-Ensemble (Multi-Objective Prompt Optimizer)

A Primer on Automatic Prompt Optimization (APO)

APO treats prompt engineering as a search problem, where the goal is to find an instruction that maximizes a model's performance on a task. Key concepts include:

- **Search Space:** The set of all possible prompts, which can be discrete (text) or continuous (embeddings).
- **Optimizer:** The algorithm that proposes new candidate prompts, which can be a traditional algorithm or another LLM.
- **Objective Function:** The metric used to score prompts, which can be single-objective (e.g., accuracy) or multi-objective (e.g., accuracy vs. latency).

Helios-Ensemble uses an optimizer LLM to navigate a discrete search space according to a multi-objective function.

Continuing the analogy, Helios-Ensemble acts as a panel of expert taste-testers and recipe improvers. It consumes the meal prepared by the chef and provides feedback to iteratively refine the recipe (the prompt) to perfection. For readers from the synthetic data community, this section explains how the generated data is consumed to produce a final, valuable artifact.

The Helios-Ensemble is the optimization engine of Project Helios, responsible for discovering prompts that are Pareto-optimal across multiple user-defined objectives. It operates on the dataset produced by Auto-Foundry, iteratively refining a population of candidate prompts to maximize their effectiveness. This section details the conceptual foundation of the ensemble, its core algorithm—a novel hybrid we term ParetoStraGo 2.0—and its approach to strategy selection and overall feasibility.

The computational complexity of the Helios-Ensemble is primarily driven by the prompt evaluation step. For a population of N prompts, a dataset of K examples, and T objectives, each iteration has a complexity of roughly $O(N \cdot K \cdot T \cdot C_{LLM})$, where C_{LLM} is the cost of a single LLM inference. The bandit-based strategy selection and prompt generation steps have negligible complexity in comparison

6.1 Conceptual Foundation: optimizer LLM

The Helios-Ensemble is built upon the emerging paradigm of using Large Language Models as optimizers themselves (Yang et al., 2024). In this framework, the optimization process is not driven by a traditional numerical algorithm but by an LLM guided by a carefully constructed meta-prompt. The meta-prompt contains the current population of candidate prompts, their performance scores across multiple objectives (e.g., accuracy, latency), and a set of strategic instructions. the optimizer LLM LLM's task is to analyze this information and generate a new, improved set of candidate prompts. This approach leverages the LLM's ability to reason about abstract tasks and perform complex, non-gradient-based optimization in the semantic space of natural language.

Algorithm 2: ParetoStraGo 2.0 Optimization Loop (Conceptual)

Input: Initial prompt population P_0, Objective functions O, Strategy set S, Population size k
Output: Pareto-optimal prompt set P^*
$P \leftarrow P_0$;
while *not converged* **do**
  Evaluate each prompt $p \in P$ on objectives O to get scores;
  Update Pareto front P^* with non-dominated prompts from P;
  Select strategy $s \in S$ using bandit algorithm;
  Construct meta-prompt with P^*, scores, and strategy s;
  Generate new candidate prompts P_{new} using Optimizer LLM;
  $P \leftarrow \text{CullToSize}(P \cup P_{new}, k)$;
end

6.2 ParetoStraGo 2.0: A Hybrid Optimization Algorithm

At the core of the Helios-Ensemble is a novel hybrid algorithm, which we term ParetoStraGo 2.0. While hybrid optimizers are an active area of research, Helios's specific synthesis of Pareto-front maintenance with explicit strategic guidance for automated signal extraction represents a novel architectural pattern. This algorithm synthesizes two state-of-the-art techniques in automatic prompt optimization: Pareto-front optimization and strategic guidance.

- **Pareto Optimization:** To handle the inherent multi-objective nature of signal extraction, the algorithm maintains a population of prompts that represent the Pareto front—the set of solutions where no single objective can be improved without degrading another (Zhao et al., 2025). This allows the system to explore the trade-offs between competing metrics like accuracy and cost explicitly.

- **Strategic Guidance:** Instead of relying solely on examples of good and bad prompts, ParetoStraGo 2.0 provides the optimizer LLM LLM with explicit, actionable strategies for improvement. This is a form of strategic guidance (Wu et al., 2024), where the meta-prompt includes not just prompts and scores, but also natural-language instructions like "try rephrasing the prompt to be more concise" or "add a constraint to handle this edge case."

This hybrid approach makes the optimization process more directed and efficient than methods that rely on unstructured search or simple example-based feedback.

6.3 Bandit-Style Strategy Selection and Feasibility

To select the most effective improvement strategy at each optimization step, Helios-Ensemble adopts the bandit-style algorithm pioneered by Ashizawa et al. (2025) to choose the most effective improvement strategy at each optimization step. The set of possible strategies (e.g., "rephrase," "add example," "simplify") are treated as arms of a multi-armed bandit. The system dynamically allocates its budget to explore different strategies, gradually learning which ones yield the greatest improvement on the Pareto front for a given task. This balances exploration of new optimization pathways with exploitation of proven ones.

The primary computational cost of the Helios-Ensemble is the evaluation of candidate prompts against the synthetic dataset. However, this process is parallel, as each prompt can be evaluated independently. The prompt generation step, performed by the optimizer LLM LLM, is sequential but constitutes a small fraction of the overall compute time. This architecture makes the Helios-Ensemble highly scalable and feasible to implement in practice, capable of efficiently searching a vast space of potential prompts to find high-performing, multi-objective solutions. This stands in contrast to earlier methods like AutoPrompt (Shin et al., 2020), DSPy's MIPRO (Opsahl-Ong et al., 2024), or soft-prompt tuning (Lester et al., 2021), which are often limited by single-objective functions and less efficient search strategies.

7 Feasibility & Illustrative Scenarios

This paper proposes a conceptual architecture for Project Helios without presenting a full experimental implementation. Therefore, this section aims to establish the project's feasibility through a series of structured thought experiments and analyses. We first conduct a cost-benefit analysis comparing the Helios pipeline to traditional manual methods. We then provide a narrative walk-through of a hypothetical scenario to illustrate the system's end-to-end operation. Finally, we map the primary technical and ethical risks to their corresponding mitigating features within the Helios architecture, arguing for the system's robustness and viability.

7.1 Cost–Benefit Analysis

To assess the economic viability of Helios, we present a comparative cost analysis against a traditional manual signal-extraction pipeline. The manual process is characterized by high, recurring labor costs for data annotation, whereas the Helios pipeline involves a higher initial computational cost that is amortized over time. As shown in Table **??**, the primary savings from Helios stem from the automation of data generation and prompt optimization, which are the most time-consuming and expensive phases of the manual workflow. The cost sensitivity of the Helios pipeline is primarily tied to the size of the text corpus and the number of signals being extracted, a relationship governed by the scaling laws of the underlying foundation models (Kaplan et al., 2020).

Table 1: Illustrative Cost-Benefit Analysis Based on Token Consumption

Phase	**Manual Pipeline Cost (Illustrative)**	**Helios Pipeline Cost (Illustrative)**
Data Generation	\$10,000 (e.g., 2,000 annotations @ \$5/ea)	\$1,500 (Auto-Foundry: 10k examples, 300M tokens @ \$5/M tokens for GPT-4o)
Optimization	\$5,000 (Engineer time for fine-tuning)	\$500 (Helios-Ensemble: 100 prompt candidates, 10 iterations, 100M tokens)
Total (per signal)	**\$15,000 (Recurring)**	**\$2,000 (Amortized Compute)**

Note: Token counts and costs are illustrative, assuming a 10k-example synthetic dataset and a moderately complex optimization run. Actual costs depend on corpus size, signal complexity, and the chosen foundation models.

7.2 Walk-Through Example: "Consequence Awareness" Signal

To make the Helios workflow concrete, we consider a hypothetical scenario: a sales organization wants to extract a "consequence awareness" signal from its sales call transcripts. This signal identifies moments when a salesperson makes a potential customer aware of the negative consequences of inaction.

1. **User Input:** The user provides a rubric: "Identify sentences where the salesperson explains the negative business impact of not adopting our solution."
2. **Initial Data Generation (Auto-Foundry):** The Generator LLM (G) produces a diverse set of initial examples based on the rubric. The Judge LLM (J) filters them for quality.
   - *Synthetic Positive:* "Without our security patch, a data breach of this nature could lead to regulatory fines exceeding \$1M."
   - *Synthetic Negative:* "Our solution offers best-in-class performance and a user-friendly interface."
3. **Initial Optimization (Helios-Ensemble):** The Optimizer LLM (O) uses this dataset to find an initial prompt, e.g., `Does the following sentence describe a negative outcome?`
4. **Hard-Case Identification:** This prompt fails on a subtle hard case from the evaluation set:

- *Hard Case:* "Your team's current workflow velocity will likely drop by another 15% next quarter if the tool integration isn't addressed." (Correct label: Yes, Predicted label: No).

5. **Feedback Loop to Auto-Foundry:** The optimizer flags this failure. G is now tasked with generating more examples of *implied* negative consequences, like the hard case. It produces new data, such as: "Sticking with your legacy system will mean your engineers continue to spend 10 hours a week on manual patching."
6. **Final Prompt Optimization (Helios-Ensemble):** Using the enriched dataset, O discovers a more robust, Pareto-optimal prompt: "`Analyze the following text. Does it state or imply a negative business consequence resulting from inaction? Answer Yes or No.`"

This example illustrates the end-to-end, self-improving nature of the Helios pipeline, moving from a simple rubric to a nuanced, production-ready prompt.

7.3 Comparative Thought Experiment

We now consider how "Stage 2" partial-automation alternatives would handle the "consequence awareness" scenario described above.

- **Alternative A (Synthetic Data Only):** A system using only a data generator like (Patel et al., 2024) would produce a large dataset based on the initial rubric. However, without the feedback loop from an optimizer, it would likely fail to generate a sufficient number of subtle, hard-to-classify examples. A human prompt engineer using this data would likely arrive at the initial, flawed prompt and be unaware of its weakness on implied consequences.
- **Alternative B (APO Only):** A system using only a prompt optimizer like (Pryzant et al., 2023) would be trained on a small, fixed dataset (likely manually annotated). While it might find a good prompt for that specific data, it would be brittle. The limited data would not represent the full range of real-world sales conversations, and the resulting prompt would fail to generalize to the unseen, hard-to-classify examples that Helios's hard-case mining loop explicitly surfaces.

This thought experiment highlights the critical advantage of Helios's tightly-coupled architecture. By unifying data generation and optimization, it overcomes the respective failure modes of partial-automation systems, leading to more robust and generalizable signal extraction.

7.4 Risk & Mitigation Mapping

Any complex AI system introduces potential risks. The Helios architecture incorporates several features designed to mitigate these risks proactively. Table 2 maps the primary risk categories to their corresponding mitigation features within Helios.

8 Discussion

This paper has articulated a conceptual architecture for Project Helios, a system designed to automate predictive signal extraction by tightly coupling synthetic data generation with multi-objective prompt optimization. In this section, we step back to discuss the broader implications of this framework. We examine its strategic value for enterprises, acknowledge its inherent limitations and ethical considerations, and outline a pragmatic roadmap for its implementation and funding.

8.1 Strategic Value for Enterprises

The primary strategic value of Project Helios lies in its potential to significantly alter the economics and velocity of signal engineering within an enterprise. The current manual paradigm is slow and expensive, creating a bottleneck that forces teams to be highly selective about which predictive signals they can afford to explore. Helios is designed to dismantle this bottleneck. By automating the most labor-intensive parts of the workflow—data annotation and prompt engineering—the framework can

Table 2: Risk and Mitigation Mapping in the Helios Architecture.

Risk Category	Specific Risk	Helios Mitigation Feature
Technical	Mode collapse in data generator, leading to low-diversity data.	Correlated sampling in Auto-Foundry to ensure breadth of examples.
	Optimizer overfits to "easy" data and fails to generalize.	Closed-loop hard-case mining forces the optimizer LLM to confront its weaknesses.
Ethical	Synthetic data amplifies societal biases present in the source corpus (Yu et al., 2023).	Independent Judge LLM can be tasked with enforcing fairness constraints defined in the rubric.
	Generated signals are used for malicious or unfair purposes.	Guardrails and multi-aspect controlled generation techniques (Zhang et al., 2024) can be integrated into the Judge module.
Operational	High compute cost of generation and optimization makes the system impractical.	Modular architecture (D6) allows for swapping in smaller, more efficient models. Cost is amortized over the signal's lifecycle.
	System becomes obsolete as foundation models evolve.	Micro-kernel design with well-defined interfaces allows for graceful evolution and component replacement.

reduce the cycle time for signal development from months to hours. This acceleration enables a much broader, more exploratory approach to feature engineering. Teams can test a vast combinatorial space of potential signals from their unstructured text data, leading to the discovery of novel, high-value predictors that would have been infeasible to investigate manually. This capability translates directly into more accurate forecasts, better-informed business decisions, and a significant competitive advantage.

8.2 Limitations

Despite its potential, the proposed Helios architecture has several important limitations that warrant discussion.

- **Dependence on Rubric Quality:** The entire Helios pipeline is initiated by a user-provided rubric. The quality of the final, optimized prompt is therefore capped by the clarity and correctness of this initial definition. A vague, ambiguous, or poorly formulated rubric will inevitably lead to the generation of a suboptimal or irrelevant signal.
- **Compute Costs:** While we argue that Helios is more cost-effective than manual annotation in the long run, the initial compute cost for generating and optimizing on a large synthetic dataset is non-trivial. The token consumption required for the Auto-Foundry and Helios-Ensemble pillars could present a significant upfront investment, potentially posing a barrier for organizations with limited computational budgets.
- **Synthetic Data Realism:** Although techniques like correlated sampling and hard-case mining are employed to enhance data quality, a gap will always exist between synthetic and real-world data. The generated dataset may fail to capture the full spectrum of linguistic nuance, domain-specific jargon, or rare events present in the true data distribution, a well-documented challenge in the field (Nadas et al., 2025).

8.3 Ethical Considerations

The automation of signal extraction introduces significant ethical considerations that must be addressed by design. The most pressing of these is the risk of **bias amplification**. Foundation models

trained on broad internet data can inherit and perpetuate societal biases related to gender, race, and other protected attributes (Yu et al., 2023). The Auto-Foundry's data generator could inadvertently concentrate these biases in the synthetic dataset, leading to the creation of unfair or discriminatory signals. The primary mitigation for this risk lies in the independent Judge LLM, which can be tasked with enforcing fairness constraints and filtering out biased content.[2]

Furthermore, the power to rapidly generate predictive signals carries a risk of **misuse**. For example, the system could be used to create signals that identify and target vulnerable populations for predatory purposes. To counter this, the Helios architecture must incorporate robust safety guardrails. Techniques for multi-aspect attribute control (Zhang et al., 2024; Yu et al., 2024) can be integrated into the Judge module to provide explicit guardrails against generating undesirable or harmful signals.

8.4 Funding and Implementation Roadmap

We envision the development of Project Helios as a staged initiative, beginning with a minimal viable product (MVP) and scaling towards a full-featured enterprise platform.

- **Phase 1: MVP and Pilot Program.** The initial phase would focus on securing funding to build a proof-of-concept. The MVP would implement the core feedback loop between Auto-Foundry and Helios-Ensemble for a single-objective optimization task. We would then partner with one or two internal teams to pilot the system on real-world forecasting problems, gathering baseline performance data and user feedback.
- **Phase 2: Generalization and Multi-Objective Expansion.** Based on the pilot's success, the second phase would focus on generalizing the framework. This includes implementing the full multi-objective capabilities of ParetoStraGo 2.0 and expanding the library of strategic guidance prompts. The goal of this phase is to create a robust, configurable platform that can serve a wider range of signal extraction tasks.
- **Phase 3: Enterprise-Wide Roll-out.** The final phase would involve deploying Helios as an internal platform for general use. This would require developing a user-friendly interface for rubric definition and objective selection, as well as comprehensive documentation and support. The long-term vision is to establish Helios as a central, central tool for data-driven decision-making across the enterprise.

9 Related Work

This section situates Project Helios within the context of several related fields. We first review the landscape of synthetic data generation and automatic prompt optimization, the two core technologies that Helios integrates. We then briefly touch upon classic prompt design techniques and the broader field of time-series foundation models to clarify Helios's unique contribution. Finally, we ground our architectural choices in established software engineering patterns.

9.1 Synthetic Data Generation

The use of generative models to create synthetic training data is a well-established technique for overcoming data scarcity. Early approaches in natural language processing, such as back-translation for neural machine translation (Sennrich et al., 2016), demonstrated the value of artificially generated data. The advent of LLMs has dramatically expanded the possibilities, enabling the generation of high-quality, human-like text for a wide range of tasks (Nadas et al., 2025). Modern frameworks like DataDreamer (Patel et al., 2024) have begun to standardize these workflows. However, a central challenge remains: ensuring the quality and diversity of the generated data. Research has shown that naive generation can lead to significant attribute bias and a lack of diversity (Yu et al., 2023), or fail to satisfy complex logical constraints (Fedoseev et al., 2024). Helios's Auto-Foundry pillar directly addresses these challenges by incorporating techniques for diversity (Kowshik et al., 2024) and using an independent judge to enforce rubric fidelity.

[2]The effectiveness of using an LLM as a fairness auditor is an active area of research and a key assumption in our proposed architecture.

9.2 Automatic Prompt Optimization (APO)

Automatic Prompt Optimization (APO) seeks to automate the discovery of the most effective natural language instructions to guide LLMs. The field has evolved rapidly, moving from gradient-based methods that tune continuous soft prompts (Lester et al., 2021) to discrete search techniques that edit textual prompts directly (Pryzant et al., 2023). A comprehensive survey of heuristic-based search methods can be found in (Cui et al., 2025). A particularly powerful paradigm is the concept of using an LLM as the optimizer LLM itself (Yang et al., 2024). Recent work in this area has focused on developing more sophisticated search strategies, including multi-objective approaches that find Pareto-optimal prompts (Zhao et al., 2025), the use of strategic guidance to make the optimization process more directed (Wu et al., 2024), and bandit-style algorithms for efficient strategy selection (Ashizawa et al., 2025). The Helios-Ensemble pillar builds directly on these advances, proposing a novel hybrid algorithm, ParetoStraGo 2.0, that synthesizes these state-of-the-art techniques.

9.3 Classic Prompt Design Techniques

The field of APO builds upon a foundation of classic prompt engineering techniques. Early work like AutoPrompt (Shin et al., 2020) demonstrated that discrete prompts could be automatically generated to elicit specific knowledge from language models. Concurrently, methods like prompt tuning (Lester et al., 2021) showed that lightweight, continuous "soft prompts" could be learned to adapt frozen LLMs to new tasks with remarkable parameter efficiency. Manual techniques, most notably chain-of-thought prompting (Wei et al., 2023), revealed that structuring a prompt to elicit intermediate reasoning steps could significantly improve performance on complex tasks. Project Helios automates and extends these foundational ideas, using an optimizer to discover complex prompt structures that go beyond simple instruction-following.

9.4 Time-Series Foundation Models

Project Helios is situated within the broader research program of building foundation models for time-series analysis. A growing body of work is focused on developing large, pre-trained models that can perform zero-shot forecasting across a wide range of domains (Ansari et al., 2024; Das et al., 2024). These models often leverage textual information to improve their predictions, using techniques like reprogramming LLMs (Jin et al., 2024) or developing autoregressive forecasters that can process natural language context (Liu et al., 2024). Other work has focused on building large-scale, multi-domain models from the ground up (Xiao et al., 2025; Shi et al., 2025). While these models are designed to *consume* textual signals, Helios addresses the critical antecedent problem of *producing* them from raw, unstructured text. The Helios framework can be viewed as an essential upstream component in an end-to-end forecasting pipeline, generating the high-quality, contextual features that models like Time-LLM or Chronos require to achieve state-of-the-art performance.

9.5 Architectural Pattern References

The design of Project Helios is not ad-hoc but is grounded in established principles of software architecture. The system's modular, two-pillar structure is an application of the micro-kernel pattern, which emphasizes building a system around a minimal core of functionality and treating other components as pluggable extensions (Bass et al., 2003). This architectural choice is motivated by the need for evolvability in the rapidly changing foundation model landscape. Furthermore, our high-level architectural view, which separates the system into Supply-Chain, Operation, and System layers, is directly adapted from the reference architecture for foundation-model-based systems proposed by (Lu et al., 2024). Adopting these established patterns provides a robust and principled foundation for the Helios framework.

10 Conclusion & Future Work

This paper has introduced Project Helios, a conceptual architecture designed to automate the extraction of predictive signals from unstructured text. We have argued that the key to unlocking this capability lies in the tight coupling of two advancing fields: synthetic data generation and automatic prompt optimization. The proposed two-pillar architecture, featuring the Auto-Foundry for zero-config data generation and the Helios-Ensemble for multi-objective prompt optimization, provides a

credible, pattern-oriented framework for building a system that can significantly accelerate the signal engineering lifecycle. By formalizing the problem, mapping out key design decisions, and grounding the architecture in established software engineering principles, we have laid the groundwork for a new class of automated, self-improving signal extraction systems.

The immediate next step for Project Helios is to secure funding and build a minimal proof-of-concept, as outlined in our implementation roadmap (Section 8.4). The initial focus will be on validating the core feedback loop between the data generator and a single-objective prompt optimizer with a small set of pilot partners. This will allow us to gather empirical data on the system's performance and refine its core algorithms.

Looking further ahead, the long-term vision for Helios extends beyond textual signals. The modular architecture is designed to accommodate multimodal inputs, allowing for the extraction of predictive signals from images, audio, and other data sources. We also envision a future where the optimization process becomes fully adaptive and real-time, with the Helios-Ensemble continuously refining prompts based on live production data. Ultimately, we see Project Helios not as a standalone system, but as a critical component in a larger ecosystem of foundation-model-powered agents, providing the essential capability of transforming raw, unstructured information into actionable, predictive intelligence.

References

Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Yuyang Wang. Chronos: Learning the language of time series, 2024. URL https://arxiv.org/abs/2403.07815.

Rin Ashizawa, Yoichi Hirose, Nozomu Yoshinari, Kento Uchida, and Shinichi Shirakawa. Bandit-based prompt design strategy selection improves prompt optimizers, 2025. URL https://arxiv.org/abs/2503.01163.

Len Bass, Paul Clements, and Rick Kazman. *Software Architecture In Practice*. 01 2003. ISBN 978-0321154958.

Wendi Cui, Jiaxin Zhang, Zhuohang Li, Hao Sun, Damien Lopez, Kamalika Das, Bradley A. Malin, and Sricharan Kumar. Automatic prompt optimization via heuristic search: A survey, 2025. URL https://arxiv.org/abs/2502.18746.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting, 2024. URL https://arxiv.org/abs/2310.10688.

Timofey Fedoseev, Dimitar Iliev Dimitrov, Timon Gehr, and Martin Vechev. Constraint-based synthetic data generation for llm mathematical reaso. In *The 4th Workshop on Mathematical Reasoning and AI at NeurIPS'24*, 2024. URL https://openreview.net/forum?id=hR4Hskr4GX.

Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y. Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and Qingsong Wen. Time-llm: Time series forecasting by reprogramming large language models, 2024. URL https://arxiv.org/abs/2310.01728.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020. URL https://arxiv.org/abs/2001.08361.

Suhas S Kowshik, Abhishek Divekar, and Vijit Malik. Corrsynth – a correlated sampling method for diverse dataset generation from llms, 2024. URL https://arxiv.org/abs/2411.08553.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning, 2021. URL https://arxiv.org/abs/2104.08691.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension, 2019. URL https://arxiv.org/abs/1910.13461.

Yong Liu, Guo Qin, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Autotimes: Autoregressive time series forecasters via large language models, 2024. URL https://arxiv.org/abs/2402.02370.

Qinghua Lu, Liming Zhu, Xiwei Xu, Zhenchang Xing, and Jon Whittle. A reference architecture for designing foundation model based systems, 2024. URL https://arxiv.org/abs/2304.11090.

Mihai Nadas, Laura Diosan, and Andreea Tomescu. Synthetic data generation using large language models: Advances in text and code, 2025. URL https://arxiv.org/abs/2503.14023.

Krista Opsahl-Ong, Michael J Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. Optimizing instructions and demonstrations for multi-stage language model programs, 2024. URL https://arxiv.org/abs/2406.11695.

Ajay Patel, Colin Raffel, and Chris Callison-Burch. Datadreamer: A tool for synthetic data generation and reproducible llm workflows, 2024. URL https://arxiv.org/abs/2402.10379.

Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. Automatic prompt optimization with "gradient descent" and beam search, 2023. URL https://arxiv.org/abs/2305.03495.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving neural machine translation models with monolingual data, 2016. URL https://arxiv.org/abs/1511.06709.

Xiaoming Shi, Shiyu Wang, Yuqi Nie, Dianqi Li, Zhou Ye, Qingsong Wen, and Ming Jin. Time-moe: Billion-scale time series foundation models with mixture of experts, 2025. URL https://arxiv.org/abs/2409.16040.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts, 2020. URL https://arxiv.org/abs/2010.15980.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. URL https://arxiv.org/abs/2201.11903.

Andrew Robert Williams, Arjun Ashok, Étienne Marcotte, Valentina Zantedeschi, Jithendaraa Subramanian, Roland Riachi, James Requeima, Alexandre Lacoste, Irina Rish, Nicolas Chapados, and Alexandre Drouin. Context is key: A benchmark for forecasting with essential textual information, 2025. URL https://arxiv.org/abs/2410.18959.

Yurong Wu, Yan Gao, Bin Benjamin Zhu, Zineng Zhou, Xiaodi Sun, Sheng Yang, Jian-Guang Lou, Zhiming Ding, and Linjun Yang. Strago: Harnessing strategic guidance for prompt optimization, 2024. URL https://arxiv.org/abs/2410.08601.

Congxi Xiao, Jingbo Zhou, Yixiong Xiao, Xinjiang Lu, Le Zhang, and Hui Xiong. Timefound: A foundation model for time series forecasting, 2025. URL https://arxiv.org/abs/2503.04118.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers, 2024. URL https://arxiv.org/abs/2309.03409.

Sangwon Yu, Changmin Lee, Hojin Lee, and Sungroh Yoon. Controlled text generation for black-box language models via score-based progressive editor, 2024. URL https://arxiv.org/abs/2311.07430.

Yue Yu, Yuchen Zhuang, Jieyu Zhang, Yu Meng, Alexander Ratner, Ranjay Krishna, Jiaming Shen, and Chao Zhang. Large language model as attributed training data generator: A tale of diversity and bias, 2023. URL https://arxiv.org/abs/2306.15895.

Chenyang Zhang, Jiayi Lin, Haibo Tong, Bingxuan Hou, Dongyu Zhang, Jialin Li, and Junli Wang. A lightweight multi aspect controlled text generation solution for large language models, 2024. URL https://arxiv.org/abs/2410.14144.

Guang Zhao, Byung-Jun Yoon, Gilchan Park, Shantenu Jha, Shinjae Yoo, and Xiaoning Qian. Pareto prompt optimization. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=HGCk5aaSvE.

A Glossary of Key Terms

Rubric A set of natural-language rules provided by a user that defines the criteria for a predictive signal.

Correlated Sampling A synthetic data generation technique that prompts an LLM to produce a portfolio of related but varied examples simultaneously to enhance diversity.

Hard-Case Mining The process of identifying examples where a model performs poorly and using them to guide further training or data generation.

Pareto Front In multi-objective optimization, the set of all solutions for which no objective can be improved without degrading at least one other objective.

Bandit Algorithm A class of algorithms for sequential decision-making that balances exploring new options with exploiting known good options.