ADO: Automatic Data Optimization for Inputs in LLM Prompts

Sam Lin[†*], **Wenyue Hua**[§*], **Lingyao Li**[‡], **Zhenting Wang**[†], **Yongfeng Zhang**[†]
[†]Department of Computer Science, Rutgers University, New Brunswick
[§]Department of Computer Science, University of California, Santa Barbara
[‡]School of Information, University of South Florida
[*]Sam Lin and Wenyue Hua contribute equally.

Abstract

This study explores a novel approach to enhance the performance of Large Language Models (LLMs) through the optimization of input data within prompts. While previous research has primarily focused on refining instruction components and augmenting input data with in-context examples, our work investigates the potential benefits of optimizing the input data itself. We introduce a two-pronged strategy for input data optimization: content engineering and structural reformulation. Content engineering involves imputing missing values, removing irrelevant attributes, and enriching profiles by generating additional information inferred from existing attributes. Subsequent to content engineering, structural reformulation is applied to optimize the presentation of the modified content to LLMs, given their sensitivity to input format. Our findings suggest that these optimizations can significantly improve the performance of LLMs in various tasks, offering a promising avenue for future research in prompt engineering. The source code is available at `https://github.com/glin2229/Automatic-Data-Optimization`.

1 Introduction

Large Language Models (LLMs) (Achiam et al., 2023; Team et al., 2023; Touvron et al., 2023) have demonstrated exceptional proficiency across a wide array of tasks. They have been successfully implemented in various real-world applications, including personalized recommendations (Xu et al., 2024; Wu et al., 2024; Hua et al., 2023), healthcare (Yu et al., 2024c,b; Li et al., 2024a), financial decision-making (Li et al., 2023b; Wu et al., 2023), and advanced language reasoning (Fan et al., 2023; Sharan et al., 2023; Jin et al., 2024a; Xu et al., 2025). In particular, LLM prompting has become a critical research area (Chen et al., 2023, 2024). This is because LLMs are highly sensitive to input content and format; even slight modifications, such as changes in word order or indentation, can significantly influence their performance (Sclar et al., 2023; Fang et al., 2024; Jin et al., 2024c).

When LLMs are employed for task inferencing, a user prompt (or query) typically comprises two primary components: a task-specific instruction and the input data to be processed according to that instruction. For example,when employing an LLM for Heart Disease classification (Baccouche et al., 2020), the task-specific instruction can be "analyze the following user's health profile to determine the likelihood of a heart attack", while the input data can include the individual's health profile, encompassing attributes such as age, medical history, and lifestyle habits. In the context of personalized recommendations, such as for beauty products (Geng et al., 2022), the instruction can be "generate beauty product recommendations based on the user's recent interaction history with other beauty products", with the input data consisting of the user's interaction history and a set of candidate beauty products to make recommendations from.

Various prompting methods have been proposed to enhance the inference performance of LLMs. For example, multiple studies have focused on crafting manual prompting strategies (Bsharat et al., 2023; Sahoo et al., 2024; Marvin et al., 2023), such as Chain-of-Thought (CoT) reasoning (Wei et al., 2022). Additionally, automated methods have been developed to search for optimal instructions tailored to specific tasks (Do et al., 2024; Li et al., 2024b). For instance, APE (Zhou et al., 2023) introduces an iterative Monte Carlo search to refine prompt instructions. Other works focus on providing in-context demonstrations (Dong et al., 2022), offering examples to guide the model's responses.

Most prior works on prompt engineering have focused on two aspects: (1) optimizing the instruction component of the prompt and (2) augmenting

Corresponding: gl550@scarletmail.rutgers.edu

Findings of the Association for Computational Linguistics: ACL 2025, pages 26134–26146
July 27 - August 1, 2025 ©2025 Association for Computational Linguistics

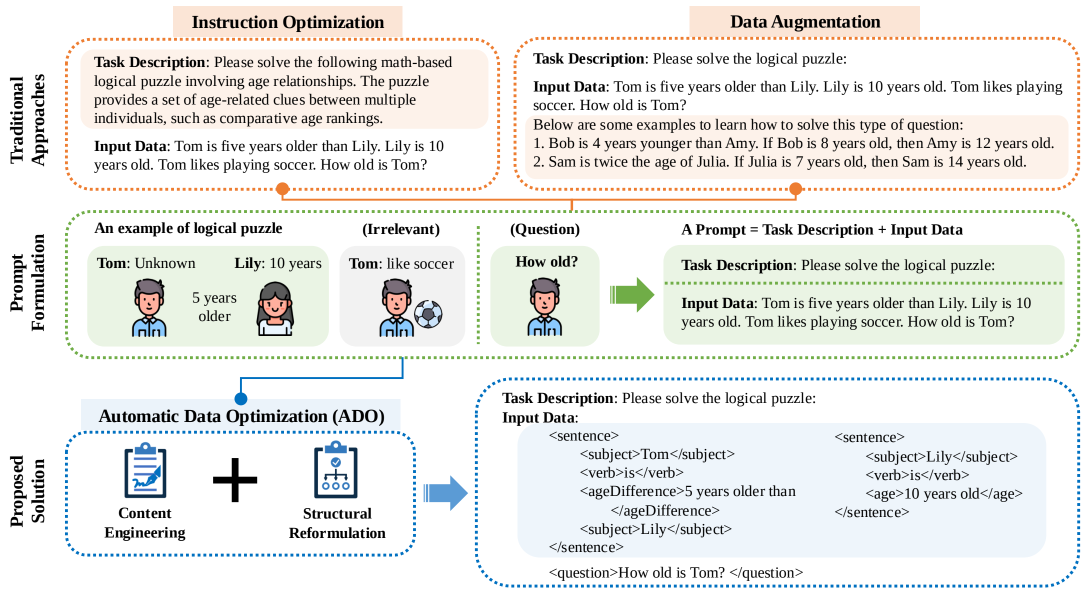

Figure 1: Types of prompt engineering approaches. Given an inference task, such as solving a logical puzzle (as shown in the middle of the figure), prior works primarily focus on either optimizing instructions or augmenting the input data with similar examples, as depicted at the top of the figure. In contrast, we propose optimizing the input data to enhance its presentation to LLMs for more effective task inference, as illustrated at the bottom of the figure.

the input data with additional context, such as in-context exemplars, as illustrated on the "Traditional Approach" section of Figure 1. Nevertheless, the role of input data optimization in enhancing LLM performance remains underexplored.

To address this gap, we investigate whether optimizing the input data portion of the prompt can also enhance performance, as depicted on the "Proposed Solution" section of Figure 1. Towards this goal, we propose a new framework "**A**utomatic **D**ata **O**ptimization (ADO)" as well as a new algorithm, "**D**iverse **P**rompt **S**earch (DPS)". This framework can optimize input data through two key strategies: content engineering and structural reformulation. First, we apply content engineering to refine input data, such as imputing missing values based on domain knowledge and removing irrelevant attributes that may hinder decision-making. Second, we leverage structural reformulation to modify the format of input data, aiming to optimize data presentation to LLMs. Together, our proposed framework has demonstrated its effectiveness to complement conventional prompting strategies to enhance LLM inference performance.

2 ADO Framework

This section outlines the objectives of input data optimization and explains the mechanisms by which the ADO framework achieves these objectives.

2.1 Framework Objective

In this work, **we conduct data optimization on the input data part of the prompt** prior to submitting the prompt to a LLM for inference. Our data optimization objectives can be categorized into two aspects: content optimization and format optimization. Content optimization emphasizes enhancing the saliency of features within the data, ensuring that the most relevant and informative attributes are highlighted. Format optimization focuses on structuring the data in an optimal format, such as tables, XML, or other representations that facilitate efficient processing and interpretation. Let $\mathbf{D}$ represents the original input data. The overall data optimization process can be considered as a combination of both content and format optimizations, resulting in an optimized dataset $\mathbf{D}'$:

$$\mathbf{D}' = f_{format}(f_{content}(\mathbf{D})) = f(\mathbf{D}) \quad (1)$$

where f is the composite optimization function. This comprehensive approach ensures that the data not only contains salient and derived features but is also presented in a format that maximizes its utility for inference tasks.

Content Optimization has been a prominent area of research across various fields and modalities (Ahmad et al., 2018; Zhou and Aggarwal, 2004). For example, in tabular datasets, where each individual is represented by a set of attribute-value

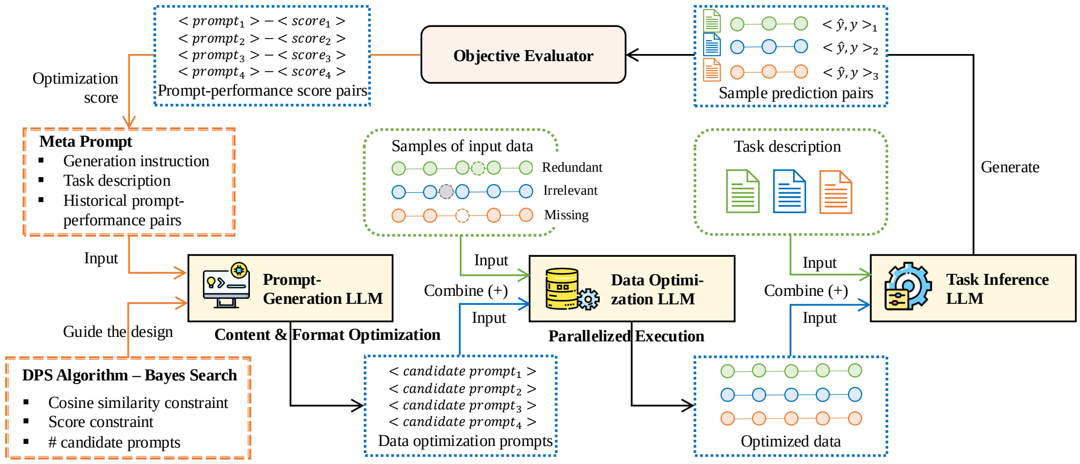

Figure 2: ADO Workflow. The Prompt-Generation LLM initially proposes task-specific instructions for optimizing input data, which the Data Optimization LLM executes on validation set samples, generating optimized inputs. These optimized samples are then processed by the Task Inference LLM to produce task predictions. The Objective Evaluator compares these predictions against the expected outputs (ground truth) using task-specific metrics to compute a score. This score represents the quality of the data optimization instructions, with prior prompt-score pairs provided as additional context to the Prompt-Generation LLM for refining instructions in future iterations.

pairs, common content optimization procedures include feature extraction, missing value imputation, and attribute aggregation (Zheng and Casari, 2018). These techniques aim to enhance the quality of the data by emphasizing salient features and reducing noise. In another example for image inputs, content optimization often involves transformations such as rotation, translation, flipping, cropping, and adjustments to brightness and contrast (Jiao and Zhao, 2019). These procedures are employed to enhance model performance by augmenting the dataset and improving the representation of important features (Barrett and Cheney, 2002; Ling et al., 2021).

Traditionally, task-specific data engineering has relied heavily on domain expertise (Ling et al., 2021). For example, in the medical field, experts may derive new attributes from existing ones—such as calculating the Body Mass Index (BMI) from weight and height measurements—to create more informative features for analysis. Similarly, for data in natural language form, such as logical puzzles or mathematical problem statements, individuals with linguistic and analytical expertise may augment the text by identifying contextual cues, deducing relevant implicit information, and explicitly defining known and unknown variables to facilitate more effective interpretation.

However, employing human experts to craft and refine each input data can be both costly and time-consuming. With recent advancements in LLMs, we propose leveraging LLMs as universal domain experts. Specifically, we investigate their ability to propose and execute content optimization procedures across datasets from diverse fields. By automating the content optimization process, we aim to transform the original dataset $\mathbf{D}$ to optimized version $\mathbf{D}$'. The objective is to reduce reliance on human expertise while maintaining or enhancing model performance. This approach not only accelerates the data preparation phase but also has the potential to uncover novel optimization strategies that may be overlooked by human practitioners.

Format Optimization concentrates on the automatic discovery of the optimal format for presenting input data to a LLM, after the content has been optimized. Recent studies have demonstrated that LLMs are highly sensitive to input formatting (Sclar et al., 2023). For example, manipulations such as positional swapping of in-context examples or alphabet shifting have been observed to influence an LLM's performance. Additionally, transforming attribute-value pairs in tabular data into structured formats like XML can enhance LLM performance on classification tasks. Similarly, converting natural language inputs into non-natural language formats using emojis, logical operators, or other symbolic figures has been shown to improve LLM performance (Lin et al., 2024a). Here, we again leverage LLM to find an optimal formatting function that maximizes the performance. By utilizing LLMs to explore various formatting strategies, we aim to identify structural reformulations that enhance the LLM's performance without altering the underlying content of the data.

2.2 Framework Workflow Design

The ADO framework employs a set of LLMs to automatically optimize the representation of input data $\mathbf{D}$. As illustrated in Figure 2, the process initiates with a Prompt Generation LLM, which proposes a data-optimization prompt $\mathbf{P}_o$ that outlines a set of procedures for modifying $\mathbf{D}$. Specifically, these procedures consist of two sequential components: the first provides step-by-step instructions for modifying the content of $\mathbf{D}$, while the second details step-by-step instructions for reformulating the content-optimized data.

Subsequently, a Data Optimization LLM progressively executes the proposed data-optimization prompt by processing both $\mathbf{P}_o$ and $\mathbf{D}$, instructing the model to generate the optimized data $\mathbf{D}'$ to implement the target function $\mathbf{D}' = f_{\text{format}}(f_{\text{content}}(\mathbf{D}))$. The optimized data $\mathbf{D}'$ is then submitted to a Task Inference LLM for processing, and its performance is evaluated on a reserved validation set, serving as the performance measure for $\mathbf{P}_o$. Finally, $\mathbf{P}_o$ and its corresponding performance are fed back into the Prompt Generation LLM as additional context, enabling it to generate improved data-optimization prompts in future search rounds.

We now formally define the ADO framework, which involves three instances of LLMs:

- Prompt Generation LLM ($\text{LLM}_\mathcal{G}$): Given a meta-prompt $\mathbf{P}_m$ used to instruct generating the data-optimization-prompt $\mathbf{P}_o$, $\text{LLM}_\mathcal{G}$ generates a set of candidate $\mathbf{P}_o$s aiming at providing instructions on how to optimize $\mathbf{D}$:

$$\mathbf{P}_o = \text{LLM}_\mathcal{G}(\mathbf{P}_m) \tag{2}$$

- Data Optimization LLM ($\text{LLM}_\mathcal{O}$): Given a data-optimization prompt $\mathbf{P}_o$, $\text{LLM}_\mathcal{O}$ optimizes $\mathbf{D}$ to produce the optimized data $\mathbf{D}'$:

$$\mathbf{D}' = \text{LLM}_\mathcal{O}(\mathbf{P}_o, \mathbf{D}) \tag{3}$$

- Task Inference LLM ($\text{LLM}_\mathcal{I}$): Using the optimized data $\mathbf{D}'$ and the task-specific instruction $\mathbf{t}$, $\text{LLM}_\mathcal{I}$ generates the final result y:

$$y = \text{LLM}_\mathcal{I}(\mathbf{D}', \mathbf{t}) \tag{4}$$

In the ADO framework, the search for the optimal data-optimization prompt $\mathbf{P}_o$ is typically conducted using a reserved set of data points $S = \{(x, y) \mid x \in \mathbf{D}_S, y \in \mathcal{Y}_{\mathbf{D}_S}\}$ where $\mathcal{Y}_{\mathbf{D}_S}$ is the set of ground truth corresponding to $\mathbf{D}_S$. Given S, we sequentially utilize the three LLM instances to generate candidate prompts $\mathbf{P}_o$s, optimize the data $\mathbf{D}$, and produce the final inference result y'. By comparing the generated outputs y and with the ground truth labels y', we can evaluate the quality of each candidate $\mathbf{P}_o$ using some task-specific loss function $L(y, y')$. The optimization of $\mathbf{P}_o$ can be formulated as minimizing the loss over S:

$$\mathbf{P}_o^* = \arg \min_{\mathbf{P}_o \in \text{LLM}_\mathcal{G}(\mathbf{P_m})} \sum_{(x_i, y_i) \in S} L(\text{LLM}_\mathcal{I}(\text{LLM}_\mathcal{O}(\mathbf{P}_o, x_i), \mathbf{t}), y_i) \tag{5}$$

Various optimization algorithms such as Automatic Prompt Engineer (APE) (Zhou et al., 2023), Automatic Prompt Optimization (APO) (Pryzant et al., 2023), and Optimization by PROmpting (OPRO) (Yang et al., 2024; Liu et al., 2024; Zhou et al., 2023) can be employed to search for a better $\mathbf{P}_o$ based on the loss function L. Nevertheless, such algorithms exhibit a potential limitation in optimizing $\mathbf{P}_o$. In the following subsection, we introduce the novel Diverse Prompt Search (DPS) algorithm to address the limitation.

2.3 DPS Algorithm for P_o Optimization

Recently, various optimization algorithms (Pryzant et al., 2023; Yang et al., 2024; Liu et al., 2024) have been proposed that leverage LLMs for automatic prompt optimization. Specifically, APE employs an LLM to propose several candidate prompts and selects the one with the best performance based on a reserved validation set. Subsequent works, such as OPRO, build upon this by directly utilizing an LLM as the prompt optimizer. For instance, OPRO instructs an LLM to iteratively propose candidate prompts, one at a time, while providing feedback on the performance of prior proposed prompts on a reserved validation set. This additional context enables the LLM to generate prompts with improved performance in subsequent iterations.

Nevertheless, recent studies (Zhang et al., 2024; Tang et al., 2024) have shown that optimizing by augmenting a single candidate prompt as context in each iteration, without any constraints on the resemblance between candidate prompts, may hinder the discovery of an optimal prompt. Despite being instructed to generate new candidate prompts that differ from previous ones, the LLM may at times converge toward semantically or lexically similar variations of prior proposed prompt(s). In our case, instead of proposing novel data optimization procedures, the LLM may keep proposing procedures that refine the wording or reorder the steps in the prior proposed procedures. This behavior reduces diversity in prompt generation, restricting exploration to a narrow region of the prompt space and yielding only marginal performance improvements.

To this end, we propose the DPS algorithm, which also employs a LLM as the prompt optimizer, while generating multiple diverse candidate prompts for each iteration of the search process, with both semantic and lexical diversity constraints enforced to grant prompt diversity. Specifically, we request $\text{LLM}_{\mathcal{G}}$ to generate k distinct candidate prompts $\{\mathbf{P}_o^1, ..., \mathbf{P}_o^k\}$ for each iteration of the search. For both semantic and lexical diversity among these prompts, we propose two constraints:

- Cosine similarity constraint (c_1): The cosine similarity between any pair of prompts should be less than c_1: $\cos(\mathbf{P}_o^i, \mathbf{P}_o^j) < c_1, \ \forall i \neq j$
- METEOR Score Constraint (c_2): The METEOR score (Saadany and Orasan, 2021) between any pair of prompts should be less than c_2: $\text{METEOR}(\mathbf{P}_o^i, \mathbf{P}_o^j) < c_2, \ \forall i \neq j$

To dynamically control the extent of prompt diversity tailored to specific tasks, we propose the novel idea of **incorporating Bayesian Search (Turner et al., 2021) to automatically determine optimal values for** k, c_1, and c_2 based on validation set performance. Since Bayesian Search has been widely employed for hyper-parameter tuning in various deep learning models, we propose to integrate this approach with automatic prompt search by treating ADO as a standalone model, with k, c_1, and c_2 as its hyper-parameters. The performance metric for each Bayesian Search iteration is defined as the highest performance achieved among all data-optimization prompts proposed by ADO with a fixed set of hyper-parameters. Such constraints ensure that the generated prompts are semantically and lexically diverse, encouraging exploration of different regions in the prompt space. For Bayesian Search details, please refer to A.1.

The generation of qualifying prompts is performed iteratively by repeatedly querying $\text{LLM}_{\mathcal{G}}$ until all k diverse prompts satisfying the above constraints are obtained. Each candidate prompt $\mathbf{P}_o^i$ is evaluated on S, based on which result we batch update the generation $\mathbf{P}_o$. The evaluation involves applying the data optimization and inference steps:

- Data optimization: $x_i' = \text{LLM}_{\mathcal{O}}(\mathbf{P}_o^i, x_i)$ where x_i is one input data in S
- Result inference: $y_i' = \text{LLM}_{\mathcal{I}}(x_i', \mathbf{t})$ where $\mathbf{t}$ is the task-specific instruction.

The performance of each candidate $\mathbf{P}_o^i$ is assessed by computing a loss function L over S:

$$l_i = \sum_{(x_i, y_i) \in S} L(y_i', y_i) \tag{6}$$

The batch of prompt-performance pairs ($\mathbf{P}_o^i$, l_i) is then appended to $\mathbf{P}_m$ to guide subsequent iterations of prompt generation. This feedback mechanism informs $\text{LLM}_{\mathcal{G}}$ about the effectiveness of previously generated prompts, enabling it to generate more promising candidates in future iterations.

By iteratively refining the set of candidate prompts and incorporating performance feedback with batch update, the DPS algorithm encourages the exploration of a broader search space. This increases the likelihood of discovering more effective data optimization procedures, ultimately enhancing the performance of the LLM on the given task.

3 Implementation Details

This section provides key implementation details of the ADO framework, including the structure of meta-prompts, the execution of parallelized data optimization tasks, and the handling of LLM hallucinations through multi-agent debate with cross-validation. By leveraging these components, the ADO framework effectively enhances both the content and format of input data to improve performance across diverse tasks while maintaining factual accuracy and efficiency.

Meta-Prompt In this purely text-based data optimization framework, the data-optimization prompt $\mathbf{P}_o$ must consist of instructions that can be executed by the LLM without relying on external tools or operations. To ensure this, we incorporate a comprehensive set of modality-specific constraints within the meta-prompt $\mathbf{P}_m$ provided to $\text{LLM}_{\mathcal{G}}$. These constraints guide the prompt generation process, ensuring that $\text{LLM}_{\mathcal{G}}$ avoids proposing optimization procedures that $\text{LLM}_{\mathcal{O}}$ is incapable of performing. For instance, when generating instructions for tabular data, the meta-prompt explicitly prohibits steps such as Principal Component Analysis (PCA), normalization, standardization, or one-hot encoding of categorical attributes, as these require tool-based operations beyond the LLM's text-based capabilities. An example of $\mathbf{P}_m$ is shown in Listing 1.

Parallelized Execution The generated data-optimization prompt $\mathbf{P}_o$ typically includes multiple procedures, each addressing a specific aspect of data engineering or reformulation (e.g., missing data imputation, structural conversion). We parse the number of procedures generated from $\mathbf{P}_o$ and employ an equivalent number of LLM instances to execute each procedure concurrently.

Parallel execution provides two advantages: (1) avoiding omission or redundancy – we observed that prompting $\text{LLM}_\mathcal{O}$ to execute a lengthy list of detailed procedures in one go often leads to omissions and repetition. By executing procedures in parallel, we mitigate these issues by breaking down the tasks into smaller, independent units of work for each LLM instance. (2) improving time efficiency – Sequential execution of a long series of procedures can be time-consuming. Since many procedures are independent of each other and can be directly applied to the raw input data, distributing them across multiple LLM instances significantly reduces the overall time required for data optimization. For procedures that depend on sequential execution – where the output of one serves as the input for the next – their execution is grouped together.

Hallucination Mitigation Instructions included $\mathbf{P}_o$ may sometimes be implemented inaccurately by $\text{LLM}_\mathcal{O}$ due to hallucinations. For example, if $\mathbf{P}_o$ includes a directive such as "Please identify the mathematical terminologies and provide concise definitions, accompanied by examples for each." $\text{LLM}_\mathcal{O}$ may generate incorrect or inaccurate definitions for some of the terms identified. These inaccuracies could mislead the performance of $\text{LLM}_\mathcal{I}$, potentially degrading overall output quality.

To mitigate the risk of hallucination and improve factual accuracy, we adapt a cross-validation method inspired by (Du et al., 2023). In this framework, we introduce an additional LLM, denoted as $\text{LLM}_\mathcal{F}$, which reviews the optimized input data to identify factual inaccuracies and provides concise explanations for any detected errors. When errors are found, $\text{LLM}_\mathcal{F}$'s feedback is passed back to $\text{LLM}_\mathcal{O}$, prompting it either to justify its original output or to agree with the corrections suggested by $\text{LLM}_\mathcal{F}$. By incorporating this cross-validation framework, we ensure a higher level of factual accuracy, leveraging the complementary strengths of multiple LLMs to reduce the likelihood of hallucinations and errors in the final output.

4 Experiments

In this section, we aim to evaluate: (1) the effectiveness of ADO as a standalone approach for performance enhancement, (2) whether DPS outperforms existing optimization algorithms in searching for data-optimization procedures, and (3) whether integrating ADO with other prompt engineering methods can further improve their performance.

4.1 Experiment Settings

Dataset To demonstrate the wide applicability of data optimization, we conduct experiments on nine publicly available, real-world datasets across various domains where LLMs are frequently applied (Fang et al., 2024; Li et al., 2023a; Lin et al., 2024b; Rouzegar and Makrehchi, 2024). These datasets include Big-Bench StrategyQA (QA) [1], Fraudulent Job Detection (Job) [2], Grade School Math 8k (GSM8k) [3], Amazon Beauty (AB) [4], Amazon Toys (AT) [5], Amazon Electronics (AE) [6], Census Income (CI) [7], Heart Disease (HD) [8], and Financial Distress (FD) [9]. For each dataset, we randomly select 1,000 samples to form the validation set S.

Modeling The evaluation modeling is twofold. First, we evaluate the effectiveness of ADO under zero-shot prompting, using three LLMs with different backbones for generalizability. To perform data-optimization procedure search, we employ APE, OPRO, and DPS algorithms. Second, we assess whether ADO can be integrated with existing prompt engineering techniques (i.e., Instruction Optimization and Data Augmentation) to further enhance their performance, with GPT-3.5 Turbo as the backbone. For Instruction Optimization, we

[1] https://github.com/google/BIG-bench/tree/main/bigbench/benchmark_tasks/strategyqa

[2] https://www.kaggle.com/datasets/shivamb/real-or-fake-fake-jobposting-prediction

[3] https://huggingface.co/datasets/DaertML/gsm8k-jsonl

[4] https://jmcauley.ucsd.edu/data/amazon/

[5] https://jmcauley.ucsd.edu/data/amazon/

[6] https://jmcauley.ucsd.edu/data/amazon/

[7] https://archive.ics.uci.edu/dataset/2/adult

[8] https://www.kaggle.com/datasets/kamilpytlak/personal-key-indicators-of-heart-disease

[9] https://www.kaggle.com/c/GiveMeSomeCredit/data?select=cs-test.csv

LLM for ADO	Algorithm	QA	Job	GSM	AB	AT	AE	CI	HD	FD	Mean
GPT-3.5 Turbo	N/A	0.578	0.619	0.285	0.124	0.129	0.211	0.788	0.617	0.639	0.443
	APE	0.575	0.633	0.721	0.161	0.184	0.241	0.839	0.687	0.658	0.522
	OPRO	0.583	0.627	0.734	**0.169**	0.195	0.238	0.846	0.681	**0.667**	0.527
	DPS	**0.589**	**0.638**	**0.755**	0.166	**0.213**	**0.253**	**0.853**	**0.704**	0.652	**0.536**
Gemini-1.5 Flash	N/A	0.569	0.607	0.299	0.137	0.115	0.197	0.791	0.625	0.612	0.439
	APE	0.581	0.621	0.698	0.159	0.176	0.219	0.827	0.701	0.661	0.516
	OPRO	0.589	0.624	0.704	0.173	0.183	**0.238**	**0.841**	0.709	0.672	0.526
	DPS	**0.595**	**0.643**	**0.729**	**0.198**	**0.201**	0.225	0.838	**0.722**	**0.699**	**0.539**
Llama-3.1 70B	N/A	0.563	0.588	0.281	0.117	0.135	0.188	0.769	0.629	0.615	0.431
	APE	0.571	0.613	0.675	0.129	0.166	0.205	0.798	0.673	0.649	0.498
	OPRO	0.574	0.619	0.693	0.135	0.173	0.213	0.806	0.692	0.657	0.507
	DPS	**0.581**	**0.635**	**0.718**	**0.159**	**0.189**	**0.229**	**0.827**	**0.711**	**0.661**	**0.523**

Table 1: ADO performance across all datasets. "LLM for ADO" denotes the LLM used within the ADO framework. "Algorithm" denotes the algorithm to search for optimal data-optimization procedures. "Mean" denotes the mean performance across all datasets. The best performance for each dataset on every LLM is highlighted in bold.

employ either Chain-of-Thought (CoT) reasoning (Wei et al., 2022) or PE2 (Ye et al., 2023) after ADO is applied; similarly, for Data Augmentation, we employ In-Context Learning (ICL) (Liu et al., 2022) subsequent to employing ADO. For CoT, we follow (Wei et al., 2022) by appending the phrase "Let's think step-by-step" at the end of the task instruction. For PE2, we employ it to search for the optimal task instruction. For ICL, we randomly select ten samples per dataset and augment them to the prompt for extra context (Liu et al., 2022). For additional modeling details, please refer to A.2.

Evaluation metrics We employ accuracy for classification tasks (with balanced accuracy for datasets with imbalanced targets) and Hit@10 for the recommendation datasets from Amazon.

Baselines To evaluate the effectiveness of ADO, we compare $\text{LLM}_{\mathcal{I}}s'$ performance without data optimization to the performance achieved after ADO is applied. To evaluate the effectiveness of the DPS algorithm on data-optimization procedure search, we compare it against two recent optimization algorithms: APE and OPRO. It is important to highlight that ADO represents a novel sub-direction in the field of prompt engineering and can be combined with existing prompt engineering techniques. Unlike a competitive relationship, ADO and techniques such as CoT, PE2, and ICL are in fact **complementary**, enabling joint application for enhanced performance. Thus, we utilize CoT, PE2, and ICL as baselines to observe whether combining ADO with any of these techniques achieves better performance compared to using them alone.

LLM Backbones We employ three instances of the same LLM as $\text{LLM}_{\mathcal{G}}$, $\text{LLM}_{\mathcal{O}}$, and $\text{LLM}_{\mathcal{I}}$. For generalizability, we test with three different LLMs, including GPT-3.5 Turbo, Gemini-1.5 Flash, and Llama-3.1 70B. Additionally, Gemini-1.5 Pro is instantiated as $\text{LLM}_{\mathcal{F}}$, which will be employed in Section 4.3. We set the temperature to 1.0 for $\text{LLM}_{\mathcal{G}}$ to encourage the generation of more creative content. For $\text{LLM}_{\mathcal{O}}$ and $\text{LLM}_{\mathcal{I}}$, we set the temperature to 0 to obtain more consistent outputs.

4.2 Result and Analysis

As demonstrated by Table 1, employing ADO for data optimization consistently leads to comparable or superior performance across all datasets on all three LLM backbones, compared to task inferencing with unoptimized data. Additionally, DPS outperforms both APE and OPRO on seven, seven, and nine out of nine datasets for GPT-3.5 Turbo, Gemini-1.5 Flash, and Llama-3.1 70B, respectively. This highlights the effectiveness of batch-based prompt search with candidates that are both semantically and lexically diverse, with the degree of diversity configured via Bayesian Search.

Furthermore, Table 2 demonstrates that integrating ADO with existing prompt engineering techniques, including CoT, ICL, and PE2, consistently results in a noticeable performance enhancement compared to employing these techniques alone, across all nine evaluated datasets. For instance, ADO significantly boosts the effectiveness of CoT, particularly in the QA, Job, and FD datasets. For QA, applying CoT alone even results in a slight performance drop compared to not applying it, while combining CoT with ADO yields substantially better performance (Figure 3 provides an additional visualization of the performance gains from ADO integration with CoT). These results demonstrate the complementarity of ADO with both Instruction Optimization and Data Augmentation methods.

Modeling variant	QA	Job	GSM	AB	AT	AE	CI	HD	FD	Mean
GPT	0.578	0.619	0.285	0.124	0.129	0.211	0.788	0.617	0.639	0.443
GPT w/ CoT	0.571	0.663	0.698	0.127	0.137	0.198	0.827	0.678	0.688	0.510
GPT w/ CoT + ADO	**0.679**	**0.807**	**0.851**	**0.185**	**0.219**	**0.257**	**0.879**	**0.751**	**0.789**	**0.602**
GPT w/ ICL	0.584	0.617	0.294	0.141	0.147	0.225	0.809	0.651	0.653	0.458
GPT w/ ICL + ADO	**0.597**	**0.641**	**0.778**	**0.199**	**0.223**	**0.262**	**0.851**	**0.728**	**0.668**	**0.549**
GPT w/ PE2	0.592	0.634	0.301	0.162	0.152	0.209	0.838	0.649	0.685	0.469
GPT w/ PE2 + ADO	**0.618**	**0.659**	**0.312**	**0.183**	**0.178**	**0.234**	**0.863**	**0.697**	**0.722**	**0.496**

Table 2: Performance when ADO is combined with other prompt engineering techniques, using GPT-3.5 Turbo as the backbone (denoted as "GPT"). "CoT + ADO" denotes applying both CoT and ADO, "ICL + ADO" denotes applying both ICL and ADO, and "PE2 + ADO" denotes applying both PE2 and ADO. For each dataset on each technique, any performance enhancement resulting from ADO integration is highlighted in bold.

4.3 Ablation Study

In this section, we perform a detailed ablation study to assess the impact of different components of the ADO framework from three perspectives: (1) whether both content optimization and format optimization are necessary, (2) whether incorporating a factual-validation LLM ($\text{LLM}_{\mathcal{F}}$) improves performance, and (3) whether data-optimizing in-context examples yields performance gains.

To examine each of these aspects, we design three corresponding experiments: (1) we explicitly constrain the data optimization process to operate solely on either content or format to observe performance changes; (2) we incorporate $\text{LLM}_{\mathcal{F}}$ into the ADO workflow for output cross-validation to evaluate its impact on task performance; and (3) we apply the same data optimization procedures to samples within the in-context examples to assess whether such alignment improves performance.

For more experimental details, please refer to A.3. The results of all three experiments are presented in Table 3 in the Appendix. As the table demonstrates, both content and format optimizations are essential for performance: removing format optimization significantly reduced performance on recommendation datasets and the CI dataset, while removing content optimization led to declines on other datasets. Moreover, incorporating $\text{LLM}_{\mathcal{F}}$ for hallucination mitigation produced comparable or improved performance across all datasets, with most significant gains on the QA, Job, and GSM datasets. Finally, optimizing input data in ICL examples led to noticeable improvements compared to its unoptimized counterpart, particularly on the Job, GSM, and FD datasets.

5 Related Work

Numerous approaches have been proposed for modifying prompts to enhance LLM performance, such as In-Context Learning and Instruction Optimization. In-Context Learning concentrates on providing the LLM with additional in-prompt exemplars from the same task domain, typically in the form of input data paired with their corresponding labels or outputs (Wei et al., 2023; Dong et al., 2022; Shin et al., 2022). This method capitalizes on the model's ability to generalize from in-prompt examples, enabling the LLM to better comprehend the expected output format and task-specific requirements based on the provided exemplars.

Instruction Optimization aims to modify the instruction part of the prompt to improve LLM performance. For example, Si et al. (2022) points out that composing better instructions can greatly boost LLM's performance on task inferencing. Wei et al. (2022) proposes CoT reasoning, which introduces immediate reasoning steps into the output generation process. As demonstrated by (Wei et al., 2022), employing zero-shot CoT substantially improve LLM performance tasks including logical reasoning, fraud detection, among many others. Extending beyond manually crafted instructions, various studies have proposed automated methods to search for optimal instructions tailored to specific tasks (Zhou et al., 2023; Pryzant et al., 2023; Yang et al., 2024). For instance, APE (Zhou et al., 2023) introduces an iterative Monte Carlo search to refine prompt instructions. It first uses an instruction-proposing LLM to generate a set of candidate instructions, then evaluates each on a validation set to select the best-performing candidates.

Despite these advances, directly optimizing the presentation of input data has received little attention. In this work, we hypothesize that optimizing both the data content and format may yield performance improvement when employing LLM for task inferencing. Building on the principles of automatic prompt optimization, we propose a novel framework called Automatic Data Optimization (ADO). In ADO, an LLM, denoted as $\mathrm{LLM}_{\mathcal{G}}$, iteratively proposes and searches data-optimization instructions aimed at maximizing LLM performance.

6 Conclusions

In this paper, we introduce a new sub-direction of prompt engineering: input data optimization, facilitated by the ADO framework and the DPS algorithm. The ADO framework automates content and format optimization by leveraging LLMs as universal domain experts, reducing the need for manual data processing. DPS enhances this process by generating diverse data optimization prompts, enabling broader exploration and increasing the likelihood of identifying optimal procedures. Empirical results demonstrate that ADO not only improves modeling performance when used alone but also further enhances performance when combined with other prompt engineering methods.

In the future, we plan to include credible task-specific factual knowledge bases to facilitate Retrieval Augmented Generations (Yu et al., 2024a), in order to further mitigate hallucination. We also aim to perform various interpretability studies under the context of input data optimization, as inspired by (Jin et al., 2024b, 2025; Sun et al., 2025).

7 Limitations

As we explore the novel approach of input data optimization within prompts, we question whether it is possible to simultaneously search for both the optimal instruction and the optimal procedures for input data optimization in a specific inference task. Currently, as detailed in the paper, we first search for the optimal data representation using ADO, and then for the optimal instruction using PE2. However, this process involves two distinct steps, and it would be more efficient to search for both the instruction and data optimization concurrently. Moreover, such a greedy, two-step search strategy may not always yield globally optimal results. Therefore, in the future, we aim to investigate the feasibility of jointly optimizing both components, as proposed in (Sordoni et al., 2024; Chen et al., 2024), to further enhance LLM performance.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Awais Ahmad, Murad Khan, Anand Paul, Sadia Din, M Mazhar Rathore, Gwanggil Jeon, and Gyu Sang Choi. 2018. Toward modeling and optimization of features selection in big data based social internet of things. *Future Generation Computer Systems*, 82:715–726.

Asma Baccouche, Begonya Garcia-Zapirain, Cristian Castillo Olea, and Adel Elmaghraby. 2020. Ensemble deep learning models for heart disease classification: A case study from mexico. *Information*, 11(4):207.

William A Barrett and Alan S Cheney. 2002. Object-based image editing. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 777–784.

Sondos Mahmoud Bsharat, Aidar Myrzakhan, and Zhiqiang Shen. 2023. Principled instructions are all you need for questioning llama-1/2, gpt-3.5/4. *arXiv preprint arXiv:2312.16171*.

Banghao Chen, Zhaofeng Zhang, Nicolas Langrené, and Shengxin Zhu. 2023. Unleashing the potential of prompt engineering in large language models: a comprehensive review. *arXiv preprint arXiv:2310.14735*.

Yongchao Chen, Jacob Arkin, Yilun Hao, Yang Zhang, Nicholas Roy, and Chuchu Fan. 2024. Prompt optimization in multi-step tasks (promst): Integrating human feedback and preference alignment. *arXiv preprint arXiv:2402.08702*.

Viet-Tung Do, Van-Khanh Hoang, Duy-Hung Nguyen, Shahab Sabahi, Jeff Yang, Hajime Hotta, Minh-Tien Nguyen, and Hung Le. 2024. Automatic prompt selection for large language models. *arXiv preprint arXiv:2404.02717*.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. 2022. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. 2023. Improving factuality and reasoning in language models through multiagent debate. *arXiv preprint arXiv:2305.14325*.

Lizhou Fan, Wenyue Hua, Lingyao Li, Haoyang Ling, Yongfeng Zhang, and Libby Hemphill. 2023. Nphardeval: Dynamic benchmark on reasoning ability of large language models via complexity classes. *arXiv preprint arXiv:2312.14890*.

Xi Fang, Weijie Xu, Fiona Anting Tan, Jiani Zhang, Ziqing Hu, Yanjun (Jane) Qi, Scott Nickleach, Diego Socolinsky, "SHS" Srinivasan Sengamedu, and Christos Faloutsos. 2024. Large language models (llms) on tabular data: Prediction, generation, and understanding - a survey. *Transactions on Machine Learning Research*.

Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. 2022. Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5). In *Proceedings of the 16th ACM Conference on Recommender Systems*, pages 299–315.

Wenyue Hua, Lei Li, Shuyuan Xu, Li Chen, and Yongfeng Zhang. 2023. Tutorial on large language models for recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems*, pages 1281–1283.

Licheng Jiao and Jin Zhao. 2019. A survey on the new generation of deep learning in image processing. *Ieee Access*, 7:172231–172263.

Mingyu Jin, Weidi Luo, Sitao Cheng, Xinyi Wang, Wenyue Hua, Ruixiang Tang, William Yang Wang, and Yongfeng Zhang. 2024a. Disentangling memory and reasoning ability in large language models. *arXiv preprint arXiv:2411.13504*.

Mingyu Jin, Kai Mei, Wujiang Xu, Mingjie Sun, Ruixiang Tang, Mengnan Du, Zirui Liu, and Yongfeng Zhang. 2025. Massive values in self-attention modules are the key to contextual knowledge understanding. *arXiv preprint arXiv:2502.01563*.

Mingyu Jin, Qinkai Yu, Jingyuan Huang, Qingcheng Zeng, Zhenting Wang, Wenyue Hua, Haiyan Zhao, Kai Mei, Yanda Meng, Kaize Ding, et al. 2024b. Exploring concept depth: How large language models acquire knowledge and concept at different layers? *arXiv preprint arXiv:2404.07066*.

Mingyu Jin, Qinkai Yu, Dong Shu, Haiyan Zhao, Wenyue Hua, Yanda Meng, Yongfeng Zhang, and Mengnan Du. 2024c. The impact of reasoning step length on large language models. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 1830–1842.

Lei Li, Yongfeng Zhang, and Li Chen. 2023a. Prompt distillation for efficient llm-based recommendation. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 1348–1357.

Lingyao Li, Jiayan Zhou, Zhenxiang Gao, Wenyue Hua, Lizhou Fan, Huizi Yu, Loni Hagen, Yonfeng Zhang, Themistocles L Assimes, Libby Hemphill, et al. 2024a. A scoping review of using large language models (llms) to investigate electronic health records (ehrs). *arXiv preprint arXiv:2405.03066*.

Yinheng Li, Shaofei Wang, Han Ding, and Hang Chen. 2023b. Large language models in finance: A survey. In *Proceedings of the fourth ACM international conference on AI in finance*, pages 374–382.

Zelong Li, Jianchao Ji, Yingqiang Ge, Wenyue Hua, and Yongfeng Zhang. 2024b. Pap-rec: Personalized automatic prompt for recommendation language model. *arXiv preprint arXiv:2402.00284*.

Guo Lin, Wenyue Hua, and Yongfeng Zhang. 2024a. Promptcrypt: Prompt encryption for secure communication with large language models. *arXiv preprint arXiv:2402.05868*.

Xinyu Lin, Wenjie Wang, Yongqi Li, Shuo Yang, Fuli Feng, Yinwei Wei, and Tat-Seng Chua. 2024b. Data-efficient fine-tuning for llm-based recommendation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 365–374.

Huan Ling, Karsten Kreis, Daiqing Li, Seung Wook Kim, Antonio Torralba, and Sanja Fidler. 2021. Editgan: High-precision semantic image editing. *Advances in Neural Information Processing Systems*, 34:16331–16345.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35:1950–1965.

Shengcai Liu, Caishun Chen, Xinghua Qu, Ke Tang, and Yew-Soon Ong. 2024. Large language models as evolutionary optimizers. In *2024 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE.

Ggaliwango Marvin, Nakayiza Hellen, Daudi Jjingo, and Joyce Nakatumba-Nabende. 2023. Prompt engineering in large language models. In *International conference on data intelligence and cognitive informatics*, pages 387–402. Springer.

Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. 2023. Automatic prompt optimization with" gradient descent" and beam search. *arXiv preprint arXiv:2305.03495*.

Hamidreza Rouzegar and Masoud Makrehchi. 2024. Enhancing text classification through llm-driven active learning and human annotation. *arXiv preprint arXiv:2406.12114*.

Hadeel Saadany and Constantin Orasan. 2021. Bleu, meteor, bertscore: evaluation of metrics performance in assessing critical translation errors in sentiment-oriented text. *arXiv preprint arXiv:2109.14250*.

Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. 2024. A systematic survey of prompt engineering in large language models: Techniques and applications. *arXiv preprint arXiv:2402.07927*.

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. 2023. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. *arXiv preprint arXiv:2310.11324*.

SP Sharan, Francesco Pittaluga, Manmohan Chandraker, et al. 2023. Llm-assist: Enhancing closed-loop planning with language-based reasoning. *arXiv preprint arXiv:2401.00125*.

Seongjin Shin, Sang-Woo Lee, Hwijeen Ahn, Sungdong Kim, HyoungSeok Kim, Boseop Kim, Kyunghyun Cho, Gichang Lee, Woomyoung Park, Jung-Woo Ha, et al. 2022. On the effect of pretraining corpora on in-context learning by a large-scale language model. *arXiv preprint arXiv:2204.13509*.

Chenglei Si, Zhe Gan, Zhengyuan Yang, Shuohang Wang, Jianfeng Wang, Jordan Boyd-Graber, and Lijuan Wang. 2022. Prompting gpt-3 to be reliable. *arXiv preprint arXiv:2210.09150*.

Alessandro Sordoni, Eric Yuan, Marc-Alexandre Côté, Matheus Pereira, Adam Trischler, Ziang Xiao, Arian Hosseini, Friederike Niedtner, and Nicolas Le Roux. 2024. Joint prompt optimization of stacked llms using variational inference. *Advances in Neural Information Processing Systems*, 36.

Guangyan Sun, Mingyu Jin, Zhenting Wang, Cheng-Long Wang, Siqi Ma, Qifan Wang, Tong Geng, Ying Nian Wu, Yongfeng Zhang, and Dongfang Liu. 2025. Visual agents as fast and slow thinkers. In *ICLR*.

Xinyu Tang, Xiaolei Wang, Wayne Xin Zhao, Siyuan Lu, Yaliang Li, and Ji-Rong Wen. 2024. Unleashing the potential of large language models as prompt optimizers: An analogical analysis with gradient-based model optimizers. *arXiv preprint arXiv:2402.17564*.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Ryan Turner, David Eriksson, Michael McCourt, Juha Kiili, Eero Laaksonen, Zhen Xu, and Isabelle Guyon. 2021. Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020. In *NeurIPS 2020 Competition and Demonstration Track*, pages 3–26. PMLR.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Jerry Wei, Jason Wei, Yi Tay, Dustin Tran, Albert Webson, Yifeng Lu, Xinyun Chen, Hanxiao Liu, Da Huang, Denny Zhou, et al. 2023. Larger language models do in-context learning differently. *arXiv preprint arXiv:2303.03846*.

Likang Wu, Zhi Zheng, Zhaopeng Qiu, Hao Wang, Hongchao Gu, Tingjia Shen, Chuan Qin, Chen Zhu, Hengshu Zhu, Qi Liu, et al. 2024. A survey on large language models for recommendation. *World Wide Web*, 27(5):60.

Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. 2023. Bloomberggpt: A large language model for finance. *arXiv preprint arXiv:2303.17564*.

Wujiang Xu, Zujie Liang, Jiaojiao Han, Xuying Ning, Wenfang Lin, Linxun Chen, Feng Wei, and Yongfeng Zhang. 2024. Slmrec: empowering small language models for sequential recommendation. *arXiv e-prints*, pages arXiv–2405.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. 2025. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110*.

Chengrun Yang, Xuezhi Wang Wang, Yifeng Lu Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. 2024. Large language models as optimizers. In *ICLR*.

Qinyuan Ye, Maxamed Axmed, Reid Pryzant, and Fereshte Khani. 2023. Prompt engineering a prompt engineer. *arXiv preprint arXiv:2311.05661*.

Hao Yu, Aoran Gan, Kai Zhang, Shiwei Tong, Qi Liu, and Zhaofeng Liu. 2024a. Evaluation of retrieval-augmented generation: A survey. In *CCF Conference on Big Data*, pages 102–120. Springer.

Huizi Yu, Lizhou Fan, Lingyao Li, Jiayan Zhou, Zihui Ma, Lu Xian, Wenyue Hua, Sijia He, Mingyu Jin, Yongfeng Zhang, et al. 2024b. Large language models in biomedical and health informatics: A bibliometric review. *arXiv preprint arXiv:2403.16303*.

Huizi Yu, Jiayan Zhou, Lingyao Li, Shan Chen, Jack Gallifant, Anye Shi, Xiang Li, Wenyue Hua, Mingyu Jin, Guang Chen, et al. 2024c. Aipatient: Simulating patients with ehrs and llm powered agentic workflow. *arXiv preprint arXiv:2409.18924*.

Tuo Zhang, Jinyue Yuan, and Salman Avestimehr. 2024. Revisiting opro: The limitations of small-scale llms as optimizers. *arXiv preprint arXiv:2405.10276*.

Alice Zheng and Amanda Casari. 2018. *Feature engineering for machine learning: principles and techniques for data scientists*. " O'Reilly Media, Inc.".

Michelle X Zhou and Vikram Aggarwal. 2004. An optimization-based approach to dynamic data content selection in intelligent multimedia interfaces. In *Proceedings of the 17th annual ACM symposium on User interface software and technology*, pages 227–236.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2023. Large language models are human-level prompt engineers. In *ICLR*.

A Appendix

A.1 Bayesian Search Specifics

Bayesian Search fits a probabilistic surrogate to the objective and chooses new hyper-parameter settings via an acquisition function that balances exploration and exploitation, yielding higher efficiency than random search (Turner et al., 2021). In this work, we propose to incorporate Bayesian Search as part of the data-optimization procedure search, by tuning k, c_1, and c_2 as "hyper-parameters" of ADO based on performance of the validation set S. This enables us to dynamically control both the number of candidate prompts to be generated per iteration for batch update, as well as the degree of diversity among candidate prompts.

A.2 Additional Modeling Specifics

When combining ADO with zero-shot CoT prompting or ICL with fixed samples, one may choose whether or not to integrate such methods into the task inference LLM (via prompt augmentation) during the search of data-optimization procedures for enhanced alignment. While such integration could potentially lead to improved performance, it also introduces greater computational overhead.

As we categorize PE2 and zero-shot CoT as two distinct prompt engineering algorithms, we constrain PE2 from producing any procedural-reasoning phrases when searching for instructions on mathematical datasets (e.g., GSM), rather than initializing the search with CoT-prompting as done in the original paper. To stay consistent with this constraint, for the GSM dataset in particular, we also explicitly specify in the ADO meta-prompt that the data optimization procedures should minimize any derivation beyond the original input data, with respect to both content and format.

Even with the constrained meta-prompt, ADO combined with PE2 still yields better performance than PE2 alone on the GSM dataset, as showcased in Table 2. For completeness, we also evaluate the standard (i.e., unconstrained) ADO paired with PE2, which achieves an accuracy of 0.811 on GSM.

A.3 Ablation Study Specifics

All experiments reported in this section are conducted with GPT-3.5 Turbo as the backbone.

Data Optimization Objectives We evaluate the effectiveness of the two optimization objectives—content optimization and format optimization—in ADO. To this end, we constrain the data-optimization prompt $\mathbf{P}_o$ to focus on either data engineering procedures (content optimization) or structural reformulation (format optimization), using zero-shot CoT as the prompting format. Specifically, we modify the meta-prompt $\mathbf{P}_m$ to explicitly prohibit instructions related to the non-evaluated aspect, ensuring $\mathbf{P}_o$ is restricted to either content or format optimization. These are denoted as "ADO-Engineering" (data engineering only) and "ADO-Reformulation" (structural reformulation only).

Factual-validation LLM We also investigate whether integrating the factual-validation LLM ($\text{LLM}_{\mathcal{F}}$) into the ADO workflow as described in Section 3 enhances performance, again with zero-shot CoT as the prompting format for the framework. Specifically, we perform cross-validation on optimized input data, iterating between $\text{LLM}_{\mathcal{F}}$ and $\text{LLM}_{\mathcal{O}}$ until a consensus is reached or a maximum of 4 rounds is completed. If no consensus reached, the optimized input from the final validation round is used for prompt construction. This configuration is referred to as "ADO w/ Factual-check."

Optimized Input for ICL In Section 4, all in-context examples are presented in their unoptimized form. Here, we examine whether optimizing the input data of ICL examples, using the same procedures applied to the evaluation data, leads to improved performance. The hypothesis is that optimized in-context examples will better align with the evaluation input data, facilitating easier implicit learning for the LLM. Thus, we optimize the ICL input data and augment the prompt with these optimized examples paired with their respective outputs, denoted as "ADO on ICL Samples."

Table 3 presents the ablation study results. For the first experiment: both data engineering and structural reformulation are crucial for maintaining performance. Limiting optimization to data engineering led to a significant drop in performance on all recommendation datasets and the CI dataset, while restricting optimization to structural reformulation resulted in performance degradation on the other datasets. For the second experiment, incorporating $\text{LLM}_{\mathcal{F}}$ for factual cross-validation yielded similar or superior performance across all datasets, with notable gains on datasets requiring factual reasoning, such as the QA, Job, and GSM. Finally, optimizing the samples within in-context examples led to noticeable improvements, highlighting the effectiveness of our alignment-based approach.

	QA	Job	GSM	AB	AT	AE	CI	HD	FD
ADO-Engineering	0.667	0.789	0.843	0.155	0.177	0.229	0.839	0.742	0.776
ADO-Reformulation	0.602	0.719	0.734	0.189	0.208	0.253	0.868	0.684	0.705
ADO w/ Factual-check	0.691	0.823	0.864	0.187	0.221	0.262	0.884	0.747	0.795
ADO on ICL Samples	0.599	0.682	0.803	0.187	0.228	0.267	0.871	0.734	0.691

Table 3: Ablation Study Performance.

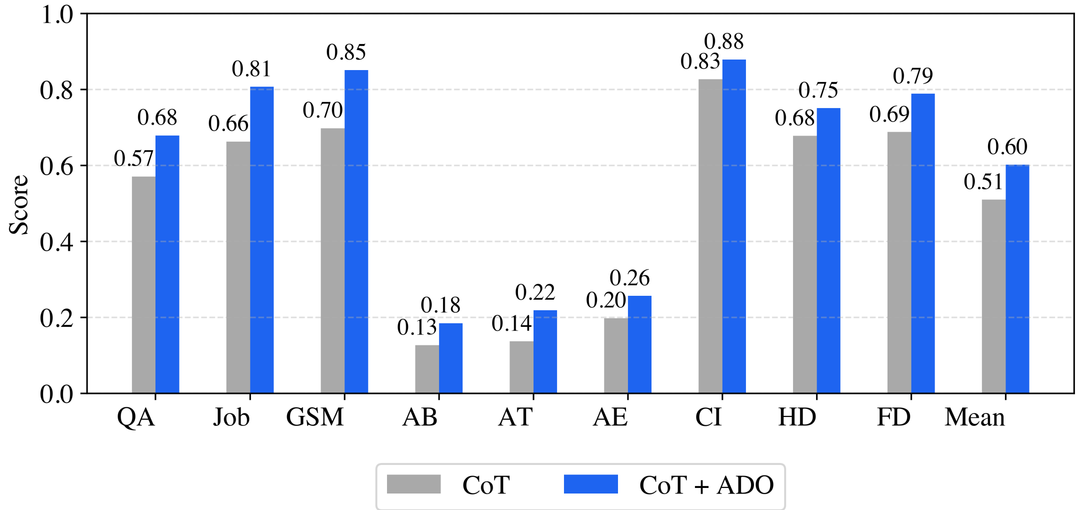

Figure 3: Performance comparison between CoT vs CoT + ADO on all datasets, with GPT-3.5 Turbo as backbone.

```
Dataset Description: <description>

Your task is to propose a creative,
detailed, and step-by-step algorithm
to enrich and then reformulate samples
in this dataset. The goal of the
algorithm is to perform thorough
data engineering and reformulation on
the sample, so that it is easier for
an LLM to generate the target outputs.

Below are some example dataset samples
with target outputs as references:

Examples:
- <sample input1>; Output: <sample output1>
- <sample input2>; Output: <sample output2>
- <sample input3>; Output: <sample output3>
- ...

Please Note:
- Do NOT refer to any external database.
- Do NOT perform vector generations.
- ONLY propose steps that an LLM
  can execute on its own.
- ...

Below is a list of prior-proposed data
optimization algorithms, provided to
you as additional context:
- Algorithm 1; Score: a1
- Algorithm 2; Score: a2
- ...
```

Listing 1: Meta Prompt Example