ADO: Automatic Data Optimization for Inputs in LLM Prompts

Anonymous ACL submission

Abstract

This study explores a novel approach to enhancing 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://anonymous.4open.science/r/ADO-6BC5/.

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 (Wu et al., 2024; Hua et al., 2023), healthcare (Yu et al., 2024b,a; Li et al., 2024a) and financial decision-making (Li et al., 2023b; Wu et al., 2023), and advanced language reasoning (Huang and Chang, 2022; Fan et al., 2023; Sharan et al., 2023).

When utilizing LLMs, the user query typically comprises two primary components: a task-specific instruction and the input data to be processed according to that instruction. For example, in the medical domain, when addressing Heart Disease classification (Baccouche et al., 2020), the task-specific instruction may be "analyze the following user's health profile to determine the likelihood of a heart attack", while the input data could include an individual's health profile, encompassing dietary and other relevant information. In the context of personalized recommendations, such as for beauty products (Geng et al., 2022), the instruction may 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. Numerous studies have demonstrated techniques to improve LLM performance by manually crafting prompts (Bsharat et al., 2023; Sahoo et al., 2024; Marvin et al., 2023), such as employing 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.

Nevertheless, most prior works on prompt engineering have focused on two primary aspects: (1) optimizing the instruction component of the prompt and (2) augmenting the input data with additional in-context examples, as illustrated in the left and middle sections of Figure 1. In this work, we investigate whether optimizing the input data portion of the prompt can also enhance performance, as depicted on the right side of Figure 1. Towards this goal, we propose a new framework "**A**utomatic **D**ata **O**ptimization (ADO)" as well as a new algorithm, aiming at optimization of data representation.

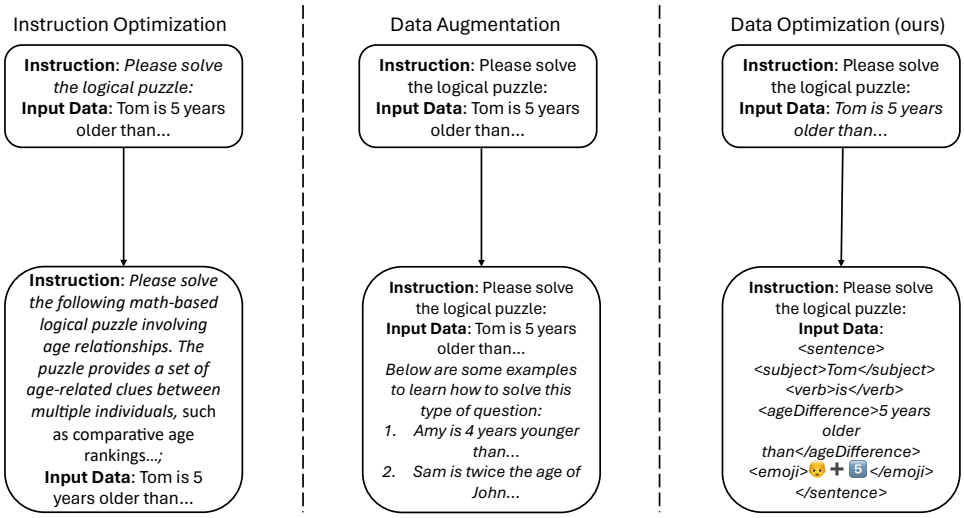

Figure 1: Types of Prompt Engineering Approaches. Prior works primarily focus on either optimizing instructions or augmenting the input data with similar examples, as depicted on the left and center of the figure. In contrast, we propose optimizing the input data itself to improve its presentation to LLMs for more effective task inference.

Specifically, we approach input data optimization from two perspectives: content engineering and structural reformulation. Content engineering may involve inputting missing values based on other attributes and removing irrelevant attributes that could distract decision-making. Following content engineering, we apply structural reformulation. As previous research has highlighted (Sclar et al., 2023; Fang et al., 2024), LLMs are highly sensitive to input format, where slight modifications such as changes in ordering or indentation can influence performance. Thus, we propose reformulating the modified content to optimize its presentation to LLMs for task inference.

The following sections are organized as below: section 2 introduces the objectives of data optimization, the ADO framework, and a new algorithm, section 3 provides implementation details, section 4 presents the experiment result, section 5 presents the relate work and section 6 concludes the paper.

2 ADO Framework

This section outlines the objectives of input data optimization and explains the mechanisms by which the Automated Data Optimization (ADO) framework achieves these improvements.

2.1 Data Optimization Objective

Data optimization objectives can be broadly categorized into two primary 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 for the task at hand. 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 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 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

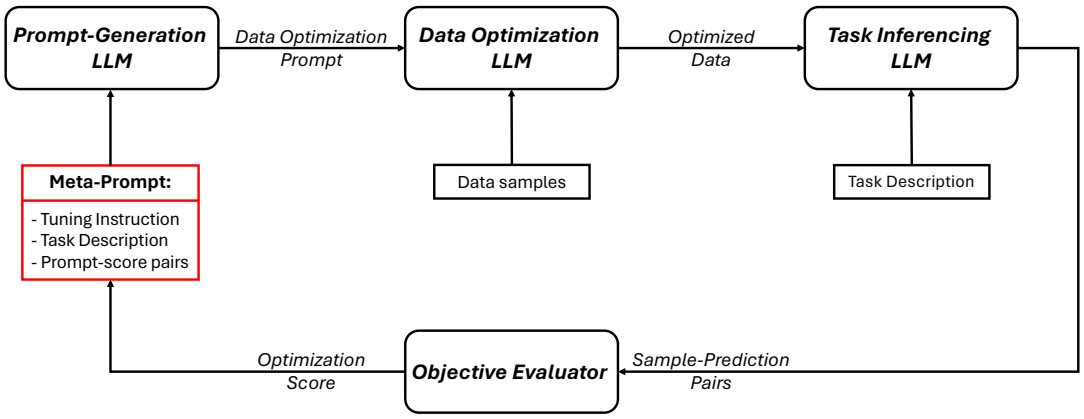

Figure 2: ADO Workflow. Initially, the Prompt-Generation LLM generates a set of task-specific instructions aimed at optimizing the input data. These instructions are then passed to the Data Optimization LLM, which applies them to data samples from a reserved validation set, producing optimized samples. The optimized data samples are submitted to the Inference LLM for generating target output predictions. The predicted outputs are evaluated against the expected outputs using task-specific metrics, and a score is computed. This score serves as a quality assessment for the previously generated data optimization instructions, and the instruction-score pair is incorporated into the meta-prompt, allowing the Prompt-Generation LLM to refine and potentially propose better instructions in subsequent iterations. The meta-prompt is highlighted in red to signify the availability of multiple prompt search algorithms that could be employed to construct it.

(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. Employing human experts to manually 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, without introducing new content. 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 ADO Framework

In this paper, we propose the ADO framework is to utilize an LLM to automatically optimize the representation of input data $\mathbf{D}$. The framework operates by searching for a data-optimization prompt $\mathbf{P}_o$ that provides guidance on how to enhance the representation of $\mathbf{D}$, implementing the target function $f_{format}(f_{content}(\mathbf{D}))$. Figure 2 presents the

workflow of ADO: guided by $\mathbf{P}_o$, the input data $\mathbf{D}$ is transformed into optimized version $\mathbf{D}'$. To generate the final answer, the LLM performs inference on the optimized data $\mathbf{D}'$ together with a task-specific instruction $\mathbf{t}$.

This framework involves 3 instances of LLMs:

- Prompt Generation LLM ($\text{LLM}_\mathcal{G}$): Given a meta-prompt $\mathbf{P}_m$ which is 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 $\mathbf{D}'$:

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

- 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. In the following subsection, we introduce the DPS algorithm for iteratively optimize $\mathbf{P}_o$.

2.3 DPS Algorithm for $\mathbf{P}_o$ Optimization

In order to search for the best $\mathbf{P}_o$ for a given task, we propose a novel LLM-based optimization algorithm, Diverse Prompt Search (DPS). In DPS, we request $\text{LLM}_\mathcal{G}$ to generate k distinct candidate prompts $\{\mathbf{P}_o^1, ..., \mathbf{P}_o^k\}$. To ensure both semantic and lexical diversity among these prompts, we enforce two specific constraints:

- Cosine similarity constraint: The cosine similarity between any pair of prompts should be less than 0.5: $\cos(\mathbf{P}_o^i, \mathbf{P}_o^j) < 0.5 \forall i \neq j$

- METEOR Score Constraint: The METEOR score (Saadany and Orasan, 2021) between any pair of prompts should be less than 0.15: $\text{METEOR}(\mathbf{P}_o^i, \mathbf{P}_o^j) < 0.15 \forall i \neq j$

These constraints ensure that the generated prompts are both semantically and lexically diverse, encouraging exploration of different regions in the prompt space.

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.

Comparison with OPRO OPRO(Yang et al., 2024) employs a similar approach in which the prompt-generation $\text{LLM}_{\mathcal{G}}$ is instructed to propose a single candidate data-optimization prompt $\mathbf{P}_o$. The performance of this candidate prompt is then evaluated on a reserved validation set S. The resulting prompt-score pair is appended to the meta-prompt $\mathbf{P}_m$, serving as a reference for generating new candidate prompts in subsequent iterations.

However, initiating the search with only one candidate prompt introduces limitations in the quest for an optimal data-optimization prompt. Despite instructing $\text{LLM}_{\mathcal{G}}$ to generate new candidate prompts that differ from those previously generated, the prompts in later iterations tend to converge toward semantically similar variations of the initial prompt. Specifically, instead of proposing novel data optimization procedures, $\text{LLM}_{\mathcal{G}}$ often refines the wording or reorders the procedures presented in the initial prompt. This behavior leads to reduced diversity in the prompt generation process and results in marginal performance gains. Due to the lack of diversity constraints, the iterative refinement in OPRO tends to explore only a limited region of the prompt space, leading to convergence towards a suboptimal solution.

In contrast, our proposed DPS algorithm addresses this limitation by generating multiple diverse candidate prompts simultaneously and enforcing both semantic and lexical diversity constraints, as previously described. By ensuring that the cosine similarity between any pair of prompts is below 0.5 and the METEOR score is less than 0.15, DPS encourages exploration across a broader region of the prompt space. This broader search increases the likelihood of identifying more effective data optimization procedures, thereby 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, 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. Below is an example of $\mathbf{P}_m$:

```
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:
- <sample1 input>; Output: <sample1 output>
- <sample2 input>; Output: <sample2 output>
- <sample3 input>; Output: <sample3 output>
- ...

Important:
- Do NOT answer the question.
- Do NOT refer to any external database.
- Do NOT perform vector generations.
- ONLY propose things that an LLM
  can do on its own.
```

Listing 1: Meta Prompt Example

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 are deployed 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.

Factual Validation 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 introduce the experiment setting including the 9 benchmarks that we experiment on and the baseline models.

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 (HR) [8], and Financial Distress (FD) [9]. For each dataset, we randomly select 1,000 samples to form the validation set S.

To assess the effectiveness of data optimization across various prompting styles, we evaluate each dataset using (1) zero-shot, (2) few-shot in-context learning, and (3) zero-shot chain-of-thought (CoT) reasoning. For few-shot in-context learning, we randomly select five samples per dataset following the approach in (Liu et al., 2022). For zero-shot CoT reasoning, we apply the method from (Wei et al., 2022), appending the phrase "Let's think step-by-step" to guide the model's reasoning process.

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

Baselines To demonstrate the effectiveness of data optimization, we compare the performance of LLMs without optimization to the performance achieved using the ADO framework. Additionally, to evaluate the effectiveness of the DPS algorithm, we compare it against the baseline prompt search algorithms, APE and OPRO. For prompt search efficiency, we implement an early stopping mechanism. After evaluating every 50 samples, the performance of the current candidate prompt is compared with the best-performing prompt identified so far. If the current prompt underperforms the best prompt for two consecutive evaluations, it is discarded, and the search continues with the next candidate. If the current prompt outperforms the best prompt across the full validation set, it is designated as the new benchmark for subsequent candidates.

[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

Prompting	Model + Search algorithm	QA	Job	GSM	AB	AT	AE	CI	HD	FD	Mean
0-shot	GPT	0.578	0.619	0.285	0.124	0.129	0.211	0.788	0.617	0.639	0.443
	GPT w/ ADO + APE	0.575	0.633	0.721	0.161	0.184	0.241	0.839	0.687	0.658	0.522
	GPT w/ADO + OPRO	0.583	0.627	0.734	**0.169**	0.195	0.238	0.846	0.681	**0.667**	0.527
	GPT w/ADO + DPS	**0.589**	**0.638**	**0.755**	0.166	**0.213**	**0.253**	**0.853**	**0.704**	0.652	**0.536**
0-shot CoT	GPT	0.571	0.663	0.698	0.127	0.137	0.198	0.827	0.678	0.688	0.510
	GPT w/ ADO + APE	0.637	0.747	0.814	0.178	0.197	0.246	0.865	0.719	0.746	0.572
	GPT w/ADO + OPRO	0.643	0.759	0.819	0.183	0.202	0.252	0.868	0.724	0.757	0.579
	GPT w/ADO + DPS	**0.679**	**0.807**	**0.851**	**0.185**	**0.219**	**0.257**	**0.879**	**0.751**	**0.789**	**0.602**
5-shot ICL	GPT	0.584	0.617	0.294	0.141	0.147	0.225	0.809	0.651	0.653	0.458
	GPT w/ ADO + APE	0.589	0.635	0.759	0.186	0.205	0.258	0.844	0.708	0.669	0.539
	GPT w/ADO + OPRO	**0.597**	0.632	0.767	0.194	0.198	**0.265**	**0.857**	0.706	0.672	0.543
	GPT w/ADO + DPS	0.592	**0.641**	**0.778**	**0.199**	**0.223**	0.262	0.851	**0.728**	**0.668**	**0.549**

Table 1: Performance across all experimental settings. For each dataset and prompting style, the best performance is highlighted in bold.

LLM Backbones We use GPT-3.5-Turbo for all LLM instances except $\text{LLM}_{\mathcal{F}}$ (employed in Section 4.3), which is instantiated by Gemini 1.0 Pro. 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 for evaluation.

4.2 Result and Analysis

As demonstrated by Table 1, applying input data optimization consistently leads to comparable or superior performance across different prompting styles, datasets, and prompt configurations (zero-shot, zero-shot CoT, and five-shot ICL), when compared to using unoptimized data. This improvement is evident across all prompt search algorithms employed. Notably, CoT prompting amplifies the effectiveness of ADO, particularly on the QA, Job, and Financial Distress (FD) datasets, where it significantly outperforms non-CoT configurations. Five-shot ICL achieves the best results for e-commerce recommendation tasks, surpassing CoT prompting in these cases.

In terms of prompt search algorithms, our proposed DPS consistently matches or outperforms both APE and OPRO across all datasets. The largest improvements are observed in the zero-shot CoT setting, where DPS shows the most significant gains, underscoring its effectiveness in optimizing prompts and enhancing overall model performance.

4.3 Ablation Study

In this section, we perform an 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.

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 investigate whether integrating the factual-validation LLM ($\text{LLM}_{\mathcal{F}}$), into the ADO workflow, as described in Section 3 enhances performance. Using zero-shot CoT, 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 four rounds is completed. If no consensus is 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 were 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

	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	0.599	0.682	0.803	0.187	0.228	0.267	0.871	0.734	0.691

Table 2: Ablation Study Performance.

the evaluation input, facilitating easier learning for the LLM. We optimize the ICL input data and augment the prompt with these optimized examples paired with their respective outputs, denoted as "ADO on ICL."

Table 2 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 validation produced comparable or improved performance across all datasets, with the most significant gains on the QA, Job, and GSM datasets. Finally, optimizing in-context examples led to noticeable improvements, particularly on the Job, GSM, and FD datasets.

5 Related Work

Numerous efforts have explored various approaches for modifying prompts to enhance LLM performance such as in in-context learning (Brown, 2020), prompt optimization, etc.

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 examples without fine-tuning, enabling the LLM to better comprehend the expected output format and task-specific requirements based on the provided exemplars.

Prompt optimization on rewriting the instruction part of the prompt in order to improve LLM performance. Si et al. (2022) points out that composing better instructions can significantly 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, optimization of input data presentation has received little attention. We hypothesize that optimizing both content features and data format can improve performance, as this allows the input to be more closely aligned with the pretraining distribution. Building on the concepts of automatic prompt optimization, we propose a novel framework called Automatic Data Optimization (ADO). In ADO, an LLM, denoted as $\text{LLM}_{\mathcal{G}}$, iteratively proposes and searches for a set of data optimization instructions aimed at maximizing LLM performance. This represents a new direction for enhancing LLM capabilities by focusing on the optimization of input data itself.

6 Conclusions

In this paper, we introduce a new direction of prompt engineering, the input data optimization, via the Automated Data Optimization (ADO) framework and the Diverse Prompt Search (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 improving the likelihood of finding optimal procedures. Together, ADO and DPS offer an efficient, salable approach to data optimization, streamlining the preparation process and improving model performance.

7 Limitations

As we introduce the novel direction of input data optimization within the prompt, we consider 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, the challenge lies in that both instruction search and data procedure search require the other component to be fixed (i.e., optimizing data procedures necessitates a fixed instruction component). In the future, we aim to explore the feasibility of jointly optimizing both components, as proposed in (Sordoni et al., 2024), to potentially 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.

Tom B Brown. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

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*.

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.

Jie Huang and Kevin Chen-Chuan Chang. 2022. Towards reasoning in large language models: A survey. *arXiv preprint arXiv:2212.10403*.

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

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.

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*.

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*.

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

Huizi Yu, Lizhou Fan, Lingyao Li, Jiayan Zhou, Zihui Ma, Lu Xian, Wenyue Hua, Sijia He, Mingyu Jin, Yongfeng Zhang, et al. 2024a. 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. 2024b. Aipatient: Simulating patients with ehrs and llm powered agentic workflow. *arXiv preprint arXiv:2409.18924*.

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. *ICLR*.