Learning from Guidelines: Structured Prompt Optimization for Expert Annotation Tasks

Wenliang Zhong[1], Haiqing Li[1], Thao M. Dang[1], Feng Jiang[1], Hehuan Ma[1], Yuzhi Guo[1], Jean Gao[1], Junzhou Huang[1]*

[1]The University of Texas at Arlington, Arlington, TX, 76019, USA

Abstract

Deep learning has significantly advanced numerous fields by training on extensive annotated datasets. However, this data-driven paradigm faces limitations such as limited adaptability and high annotation costs, particularly when precise adherence to detailed, domain-specific guidelines is required in annotation. This challenge raises a critical question: Can models effectively shift from data-driven learning to autonomously leveraging guidelines with minimal annotated examples? To address this, we propose the Guideline-Driven Prompt (GDP) optimization framework, which shifts the learning paradigm from data-driven training to guideline-driven reasoning. GDP leverages Retrieval Augmented Generation (RAG) to retrieve essential fragments from complex guidelines and synthesize them into structured, executable prompts. A tree-based optimization algorithm systematically constructs and refines these prompts, explicitly capturing the intricate logic embedded in professional guidelines through a latent pipeline structure. Empirical evaluations on four datasets ranging from diverse domains and different tasks demonstrate that GDP effectively transitions the learning process from data-intensive methods to a guideline-driven approach in tasks requiring detailed and complex guideline adherence, reducing dependence on extensive annotated datasets.

Project Page — https://guideline-driven-prompt.github.io/

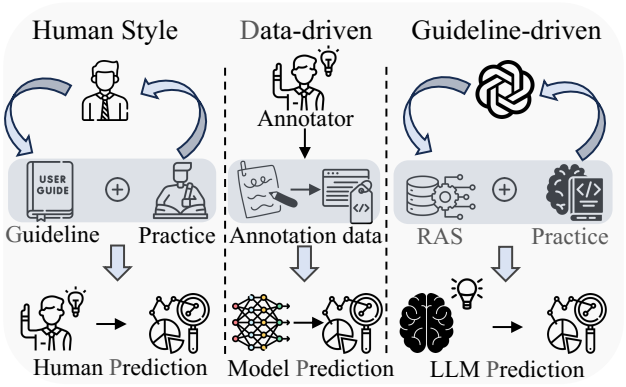

Figure 1: Comparison with different learning styles: Human-style learning mainly learns from textbooks and consolidates the understanding with practice; Data-driven learning mainly uses large-scale annotated data to update model parameters, which serves as the storage of knowledge. Guideline-driven learning is similar to human-style learning. It stores guidelines in some knowledge bases, gains fundamental understanding of the guideline, and reinforce the understanding through few-shot practice.

Introduction

Deep learning (DL) has made significant progress over the past decade by training models on large, task-specific collections of human-labeled data (Kumar et al. 2024). These models learn by minimizing an objective on massive annotated examples and later use the learned parameters to predict labels for new inputs. While this workflow achieves high accuracy, it poses two major drawbacks: (1) each new task, dialect, or label schema requires an expensive round of annotation, and (2) annotation demands considerable human effort, requiring annotators to follow *task-specific guidelines* strictly to ensure correctness. Recent advancements of Large Language Models (LLMs) (Achiam et al. 2023; Team et al. 2023) have demonstrated that they can perform general tasks without task-specific training, which offers a promising solution to mitigate these challenges. However, current LLMs still desire annotated data for domain-specific tasks to achieve optimal performance, which is either through prompt optimization (Wang et al. 2023b; Ashok and Lipton 2023) or model fine-tuning (He et al. 2023; Zhao, Dang, and Grover 2023). This limitation raises a new question: *"Can an LLM autonomously learn from guidelines with minimal annotated examples to perform domain-specific tasks?"*

Addressing this question is crucial for enabling autonomous learning in LLMs in real-world scenarios. Rather than relying entirely on data-driven supervision, models could learn in a manner similar to humans, acquiring core knowledge from textbooks (i.e., guidelines) (Karpatne, Jia, and Kumar 2024) and reinforcing that understanding with only a small number of practice examples. This paradigm also enhances the flexibility of LLMs, as updates to the guideline would no longer require costly reannotation efforts. However, guideline-based learning introduces several

*Corresponding author: jzhuang@uta.edu

challenges. First, real-world tasks often involve complex guidelines that are difficult for an LLM to interpret in their raw form (Wang et al. 2023a). Second, these guidelines vary in form and structure, so a general and unified representation format is required (Zhang et al. 2024). Third, unlike supervised learning, guideline-based learning cannot rely on loss functions or gradient updates to encode domain knowledge into model parameters (Ashok and Lipton 2023). Therefore, it is essential to develop an effective representation that allows LLMs to understand and apply guideline information during inference.

Existing methods only partially mitigate these difficulties. Approaches that compress entire guidelines into task definitions as flat prompts often discard critical contextual details (Ashok and Lipton 2023; Wang et al. 2023a). Strategies that enrich prompts with reasoning-based demonstrations add details, yet their effectiveness depends on the quality and coverage of examples (Wei et al. 2022). Fine-tuning LLMs to internalize guideline representations reintroduces the very annotation cost that guideline-based learning seeks to avoid (Sainz et al. 2023; Zhao, Dang, and Grover 2023; Gunasekar et al. 2023). More related works are discussed in Appendix D. Therefore, a framework that enables LLMs to learn primarily from guidelines remains unexplored.

In this paper, we introduce a Guideline-Driven Prompt optimization (GDP) framework to tackle the challenges outlined above. To handle complex real-world guidelines, we adopt the paradigm of Retrieval Augmented Generation (RAG) (Lewis et al. 2020) by storing the guidelines as a knowledge base. Yet, the information retrieved through RAG is inherently local, making it difficult for LLMs to capture the full semantics and structure of the guideline as a whole (Petroni et al. 2020). To overcome this limitation, we allow the LLM to learn from retrieved fragments and synthesize them into a prompt, which is then optimized to produce a faithful and effective representation of the guideline. We then replace flat free-form text prompts with a structured prompt format to enhance the model's understanding further. Our analysis reveals that many guidelines can be naturally modeled as procedural workflows, which align well with Chain-of-Thought (CoT) prompting strategies (Wei et al. 2022). Building on this insight, we introduce a tree-search algorithm (Yao et al. 2023) that iteratively refines the prompt, enabling the LLM to progressively deepen its understanding of the guideline and ultimately converge on an optimal structured prompt.

We summarize contributions as follows: (1) We introduce the first guideline-driven prompt optimization paradigm that reduces reliance on large annotated datasets by enabling LLMs to follow domain-specific instructions with minimal supervision. (2) We propose a structured prompt construction method that integrates guideline knowledge using Retrieval Augmented Generation to capture both local and global context. (3) We design a tree-search algorithm that refines structures and contents of prompts as procedural pipelines, enhancing alignment with Chain-of-Thought reasoning. (4) We demonstrate the effectiveness of our approach on four guideline-intensive tasks across various domains, showing improved performance and adaptability.

Preliminary

Problem Formulation

In this section, we formally define the problem across three paradigms, demonstrating that classical methods depend on extensive annotations and ignore valuable guidance, while our framework is tailored to mimic the human learning pipeline under resource-limited conditions.

Human-style learning (Fig.1 left). A learner first studies a concise textbook $\mathcal{S}$ and then consolidates understanding by solving a mock exam $X_k = \{(x_i, y_i)\}_{i=1}^{k}$, where each (x, y) is a question–answer pair. For a new question x_q, the learner answers by $\hat{y}_q = \psi\big(U(\mathcal{S}),\, x_q\big)$, where $U(\mathcal{S})$ is the learner's internalised understanding of the textbook and ψ denotes human reasoning. If the answer is incorrect ($\hat{y}_q \neq y_q$), a self-review mechanism $Rev(\cdot)$ updates that understanding using the failure case as: $U(\mathcal{S}) \leftarrow Rev(\hat{y}_q, y_q, \mathcal{S}, U(\mathcal{S}))$. Because the student relies primarily on the book $\mathcal{S}$, only a small number of examples in X_k is required, keeping annotation effort *minimal* (i.e., k is small).

Data-driven learning (Fig.1 middle). Denote $\mathcal{X}$ and $\mathcal{Y}$ as the input and target spaces of a domain-specific task T. Annotators first create a large task-specific training set $X_M = \{(x_i, y_i)\}_{i=1}^{M} \subset \mathcal{X} \times \mathcal{Y}$, where $M \gg 1$. The model $f_\theta : \mathcal{X} \to \mathcal{Y}$ is fit by empirical loss minimization: $\theta^* = \arg\min_\theta \frac{1}{M} \sum_{i=1}^{M} l\big(f_\theta(x_i), y_i\big)$, $l(\cdot)$ is any task-appropriate loss function. Finally, for any new query x_q, the trained model returns the prediction $\hat{y}_q = f_{\theta^*}(x_q)$.

We argue that classical data-driven methods do not fully mirror the human learning process. In particular, those systems substitute the human reasoning ψ with a parameterized model f_θ trained on a **large** labeled dataset. Since stochastic-gradient methods update θ using only the input–label pairs (x, y), they have no mechanism to incorporate the explicit guidelines $\mathcal{S}$. Consequently, the model learns solely from data and entirely **discards** the review step $Rev(\cdot)$ that humans use to enhance their understanding.

Guideline-driven learning (Fig.1 right). Let $\mathcal{S}$ be the textual guidelines that specify how to perform T. A frozen LLM $\mathcal{L}_\theta$ maps a prompt–query pair to a prediction as follows: $\mathcal{L}_\theta(\texttt{prompt}\ \ P,\, x_q) \to \hat{y}_q \in \mathcal{Y}$. We assume *very-low supervision*, a support set $X_k = \big\{(x_i, y_i)\big\}_{i=1}^{k} \subset \mathcal{X} \times \mathcal{Y}$, is the only annotated data available and $k \ll M$. The parameters θ remain fixed (no fine-tuning). Our goal is to discover a prompt P^* that initializes from the LLM's fundamental understanding of the guideline and encodes the guideline with maximal predictive power, such that $s(P^*; X_k) = \max_P s(P; X_k)$, where s is empirical accuracy of prompt P on X_k: $s(P; X_k) = \frac{1}{k} \sum_{(x,y)\in X_k} \mathbb{I}\Big[\arg\max_{\hat{y}} \mathcal{L}_\theta(P, x) = y\Big]$. To align with the human learning paradigm, we introduce a guideline-driven framework in which an LLM $\mathcal{L}_\theta$ plays the role of the human reasoning ψ. While the optimization module emulates $Rev(\cdot)$ by retrieving the most relevant information from $\mathcal{S}$ and assembling them into the optimal prompt P^*. Details of our proposed method are provided in the following section.

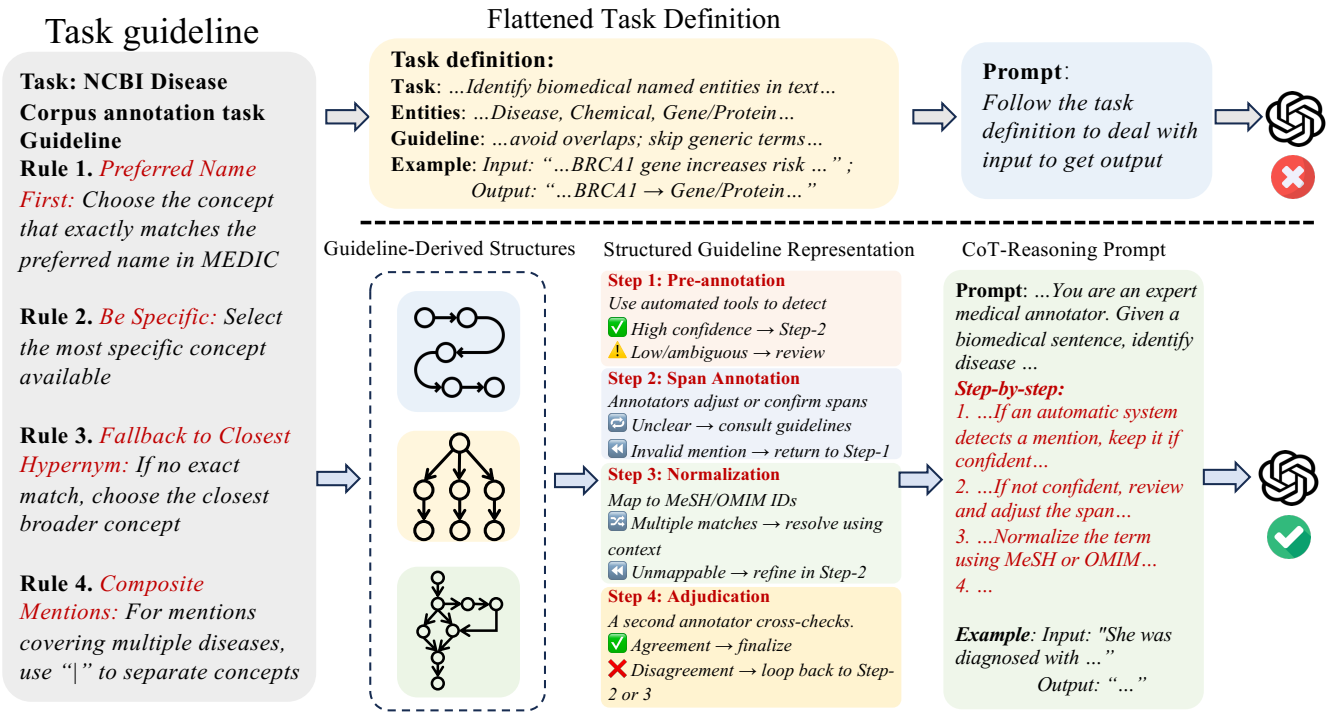

Figure 2: Guideline Representation. Real-world guidelines may contain complex logic and redundant information. Existing works that simplify the guideline as a task definition neglect the structural relations and detailed information. On the contrary, we find that most guidelines can be represented as pipelines derived from three abstract structures. By abstracting guidelines as pipelines, we effectively model the guideline in a CoT prompting format.

Processing Guidelines as Knowledge Bases

In domain-specific annotation tasks, the guideline S is often complex and presented as PDF files, websites[1], etc. These guidelines typically consist of multiple steps or subprocedures that define how decisions should be made under various conditions (Sainz et al. 2023). Depending on the domain, S may describe a linear workflow or a collection of loosely ordered subtasks. We summarize that most guidelines can be represented in three abstract forms of structures: sequential, parallel, or hybrid (shown in Figure 2). Details are discussed in Appendix A.

Feeding the entire guideline directly into the prompt is impractical for two key reasons (Ashok and Lipton 2023; Wang et al. 2023a): (i) it may exceed the input context limitations of LLMs, and (ii) it often overwhelms the model with irrelevant or overly specific information. To address this challenge, we adopt a retrieval-augmented approach that enables selective access to relevant parts of the guideline during prompt construction and optimization. Specifically, we segment S into smaller components and encode them as dense vectors via an encoder, which are then stored in a Retrievable Augmented System (RAS) $\mathcal{R}$ (Lewis et al. 2020). Given a query either from the input or from a prompt refinement operation, $\mathcal{R}$ retrieves the most relevant guideline segments using semantic similarity in the embedding space. This design allows us to treat the guideline as a structured knowledge base, from which specific instructions can be retrieved dynamically. Formally, given a query q , we retrieve the most similar chunks $\{c_1, c_2, \cdots, c_l\}$ from S such that: $\text{similarity}_i = \arg\max_{c \in S}(\text{sim}(e_q, e_c))$, where e_q and e_c denote the embedding vectors of the query and a candidate segment, and $\text{sim}(\cdot, \cdot)$ is a similarity metric such as cosine similarity.

While it may seem natural for LLMs to use RAS directly for inference, the retrieved guideline fragments are often too local or incomplete to support full task comprehension. Instead, we reframe the role of RAS as an active controller and coordinator within the prompt optimization process. Instead of feeding retrieved text directly into the model, RAS is used for two sides. First, it provides the inference LLM a fundamental summary of the guideline, similar to human learner's initial understanding of the textbook. Second, it is used to iteratively supplement and revise the prompt during training on few-shot examples. This mechanism allows the model to construct a globally coherent and task-aware prompt that reflects the structure and semantics of the full guideline in a format that is interpretable and effective for LLM inference.

Representing Guidelines as Structured Prompts

A central challenge is how to represent the guideline in a way that is both faithful to its details and interpretable by an LLM. One intuitive solution is to condense the entire

[1]See NCBI disease annotation for example: https://www.ncbi.nlm.nih.gov/CBBresearch/Dogan/DISEASE/Guidelines.html

guideline into a single paragraph of task definition within a prompt (Ashok and Lipton 2023). However, such summarizations often lose critical information, especially the fine-grained subtasks and the dependencies between them. To better preserve the guideline's procedural structure, as shown in Figure 2, we model the process as a pipeline composed of discrete steps. Each step corresponds to a subtask drawn from the original guideline, and the pipeline specifies the order in which these steps should be executed. Notably, this order does not need to match the layout of the original guideline because many real-world guidelines contain loosely ordered instructions, and enforcing a fixed order may hinder generalization. Instead, we treat the execution order as an optimization target adaptable to inputs and driven by performance (Zhang et al. 2024). Under this formulation, a flat task definition becomes a special case of the pipeline, where all logic is compressed into a single step. In contrast, a multi-step pipeline enables explicit modeling of intermediate decisions, modular reasoning, and flexible adaptation to complex guidelines.

To encode such pipelines in prompts that are interpretable and executable for LLMs, we adopt a step-by-step format inspired by Chain-of-Thought prompting (Wei et al. 2022). The prompt first lists the procedure as a series of steps, followed by an example that illustrates how each step is applied sequentially. This format supports diverse guideline structures, and avoids the overhead of treating each subtask in isolation or including the entire guideline verbatim. It also improves efficiency by guiding the LLM through structured reasoning rather than overwhelming it with undifferentiated instructions. Designing an effective pipeline structure, however, remains a non-trivial task. To this end, we propose a tree-based search algorithm that systematically explores and refines pipeline representations to maximize annotation performance. We describe this algorithm in the next section.

GDP Tree-Guided Optimization Framework

We present a tree search algorithm for optimizing guideline-aware prompts. Inspired by Monte Carlo Tree Search (MCTS) (James, Konidaris, and Rosman 2017; Zhang et al. 2024), our method explores alternative pipeline structures by decomposing the guideline into modular steps. Unlike previous approaches that flatten the guideline into a single task description (Ashok and Lipton 2023; Wang et al. 2023a), we represent the prompt as a step-by-step pipeline that mirrors the multi-stage reasoning process of human annotators. The goal of the algorithm is to discover an effective pipeline structure tailored to the target task and dataset. Since the optimal decomposition is not known in advance, the search process begins from a generic initial prompt that summarizes the task at a coarse level. At each iteration, the algorithm proposes and evaluates candidate extensions, either by splitting existing steps into finer subtasks or by refining the content of the current pipeline. The most promising extension is selected and further explored. We describe the components of the algorithm in the following paragraphs. Pseudocode is provided in Algorithm 1.

Algorithm 1: GDP: Tree-Guided Prompt Optimization (Simplified. Full Version in Appendix)

Require: Initial prompt P_0, training set X, max iterations T, RAS $\mathcal{R}$; hyper-parameters: Top-K and early-stopping ES
Ensure: Optimized prompt P^*

```
Initialize tree and candidate list with root node from P_0.
γ ← 0                                              ▷ early-stopping counter
for t = 1 to T do
    Select a node from the candidate list by score-based sampling.   ▷ Selection
    Generate two prompts from the selected node via R with pipeline decomposition and content optimization.   ▷ Expansion
    Create two new nodes from each prompt using X.
    Compute scores and update the candidate list with K-pruning.   ▷ Evaluation
    if any new node outperforms the previous best then
        P* ← best new prompt;   γ ← 0
    else
        γ ← γ + 1
    end if
    if γ ≥ ES then
        break                                      ▷ early stopping
    end if
end for
return P*
```

Initialization The tree search algorithm begins with providing the LLM the fundamental knowledge of the guideline through an initial prompt P_0 summarized by the RAS $\mathcal{R}$.

We find that including a feasible example, which is typically drawn from the guideline itself, effectively improves the adherence of LLMs to the expected output format, which can be complex and difficult to describe explicitly in certain tasks. The root node N_0 of the search tree $\mathcal{T}$ is initialized with P_0. From this root, the algorithm incrementally derives more structured pipelines by extending the general task description into finer-grained stages. Each new node in the tree contains a revised prompt and is evaluated by performing inference on the training samples. The resulting score is used to guide subsequent decisions in the search process.

$$P_0 = \mathcal{R}(\text{"Summarize } S\text{"}) \tag{1}$$

$$N_0 = \text{CreateNode}(P_0, X) \tag{2}$$

Each node N_i is initialized as:

$$N_i = (P_i, s_i, F_i, r_i) \tag{3}$$

where P_i is the prompt, s_i is the average inference score of the training samples. In addition, all failed predictions are saved as F_i and error information (reasons) is stored as r_i.

Selection At each iteration, the algorithm selects one node from the candidate list $\mathcal{C}$ to expand by generating its child nodes. This selection is guided by the node's inference score. To encourage both exploitation of high-scoring nodes and exploration of underexplored ones, we use a probabilistic

sampling strategy that mixes two distributions: a softmax over the scores and a uniform distribution over all candidates. The sampling probability for node i is given by:

$$p_i = \lambda \cdot \frac{1}{n} + (1 - \lambda) \cdot \frac{\exp(s_i)}{\sum_{j=1}^{n} \exp(s_j)} \quad (4)$$

Here, s_i is the inference score of node i, n is the number of current candidates, and $\lambda \in [0, 1]$ controls the balance between exploration and exploitation.

Unlike traditional tree search algorithms that restrict selection to children of a single parent node (Wang et al. 2023b), we maintain a global top-K candidate list across the entire tree. This list includes the K highest-scoring nodes observed so far. Each time a new node is created, its score is evaluated, and the list is updated accordingly. This strategy avoids spending resources on low-quality branches while still preserving sufficient diversity in the search process.

Denote the resulting probability distribution as $\mathcal{D} = [p_1, p_2, \cdots, p_n]^T$. A node is sampled from the candidate list following the categorical distribution $N_{cur} \sim \text{Categorical}(\mathcal{C}|\mathcal{D})$ for extension.

Extension The extension phase is responsible for generating new prompt candidates by modifying the current pipeline. Unlike prior prompt optimization methods that mainly focus on tweaking expressions or adding details based on failure cases (Sainz et al. 2023; Ashok and Lipton 2023; Wang et al. 2023b; Zhu et al. 2023), our framework aims to revise both the structure and content of the prompt in a principled way. To this end, we introduce two complementary strategies: one that adjusts the pipeline structure by decomposing coarse tasks into finer subtasks, and another that refines the content to better address observed errors.

The first strategy, **pipeline decomposition**, targets structural refinement. The RAS $\mathcal{R}$ is asked to decompose the current step into a sequence of smaller subtasks, each with its own instruction, which are appended to the prompt. This transformation allows the LLM to process annotation decisions in a more modular and transparent way. The included example is modified to reflect the updated step-by-step structure while input-output formats are preserved.

$$P_{dec} = \mathcal{R}(N_{cur}.P, N_{cur}.F, N_{cur}.r, \mathcal{I}_{cur}; \delta_{\text{structure}}) \quad (5)$$

$$N_{dec} = \text{CreateNode}(P_{dec}, X) \quad (6)$$

Here, $\delta_{\text{structure}}$ indicates the optimization instruction for decomposing the existing pipeline for finer steps. $\mathcal{I}_{cur}$ indicates the historical trajectory of N_{cur}, which is aggregated from prompts and error information of its ancestry nodes.

The second strategy, **content optimization**, focuses on improving the clarity and correctness of the current prompt content without altering its structural granularity. The RAS $\mathcal{R}$ analyzes model errors, identifies the major failure, and adjusts the prompt accordingly to mitigate that issue.

$$P_{opt} = \mathcal{R}(N_{cur}.P, N_{cur}.F, N_{cur}.r, \mathcal{I}_{cur}; \delta_{\text{sem}}) \quad (7)$$

$$N_{opt} = \text{CreateNode}(P_{opt}, X) \quad (8)$$

Here, δ_{sem} indicates the instruction for optimizing the existing pipeline for details. Finally, both N_{dec} and N_{opt} are used for updating the candidate list.

$$\mathcal{C} = \text{TopK}(\mathcal{C} \cup \{N_{\text{dec}}, N_{\text{opt}}\}, K) \quad (9)$$

All initialization and extension prompts used in GDP are in Appendix H. In experiments, we find that in early stages of the search, pipeline decomposition is often favored to introduce coarse-to-fine structure. In deeper levels, content optimization becomes more effective as the pipeline matures. For each extension, the RAS $\mathcal{R}$ integrates inference feedback from the current node and its ancestors, allowing new prompts to benefit from prior errors and corrections.

Backtracking To ensure that prompt optimization is grounded in actual model behavior, we introduce a backtracking mechanism that analyzes failure cases and uses them to guide future extensions. Each incorrect prediction is independently sent to the RAS $\mathcal{R}$, which is tasked with identifying the underlying error. These instance-level explanations are then aggregated into a single summary that captures the major failure reason at the current node. It serves as a high-level diagnostic signal for prompt refinement.

Evaluation and Termination The search process terminates when a maximum number of iterations is reached or when the performance fails to improve over a predefined number of consecutive iterations. The prompt corresponding to the highest-scoring node in the search tree is selected as the final output and used for evaluation on the test set.

Experiments

Experimental Setup

Tasks and Datasets We evaluate GDP in two guideline-intensive information-extraction tasks on four datasets across diverse domains: Named Entity Recognition (NER) and Event Extraction (EE). For NER, we consider three datasets: NCBI-Disease (clinical disease recognition) (Doğan, Leaman, and Lu 2014), WNUT (emerging entity recognition in news) (Derczynski et al. 2017), and HarveyNER (fine-grained location tagging for disaster contexts) (Chen et al. 2022). For EE, we use CASIE (Satyapanich, Ferraro, and Finin 2020) for cybercrime event detection. Guidelines for these datasets are obtained from public documentation or accompanying publications.

Following the setup in Section 3.1, we focus on few-shot scenarios where only limited training data is available mirroring the human annotation setting. Specifically, we evaluate performance using $k = \{5, 10, 20\}$ training examples randomly sampled from the original training sets. Prompts are optimized on these small subsets and evaluated on the full test sets. Each experiment is repeated with three random seeds, and the average results are reported. Moreover, in few-shot scenarios, In-Context Learning (ICL) (Dong et al. 2022; Min et al. 2022) is another technique known to be effective. Besides the default 1-shot setting, we also experiment with the 5-shot setting where examples are from the training subset. Additional dataset details, including guideline structure and data splits, are provided in Appendix B.

Baselines We compare GDP against a range of prompt-based baselines: (a) Standard Prompting (Standard) (Ashok and Lipton 2023; Wang et al. 2023a): Directly uses summary prompts based on the guideline; (b) RAS Prompting (Lewis et al. 2020) (RAS-P): Retrieves relevant guide-

Baselines	ICL	GPT-3.5-Turbo					GPT-4				
		NCBI	WNUT	Harv.	CASIE	Avg	NCBI	WNUT	Harv.	CASIE	Avg
Standard Prompting†	1-shot	33.3	41.4	27.2	69.6	42.9	39.2	47.6	39.1	76.2	50.5
	5-shot	45.0	39.6	26.1	74.8	46.4	56.8	50.1	35.6	80.6	55.8
RAS Prompting†	1-shot	35.5	40.4	25.6	70.0	42.9	39.9	48.4	41.0	78.3	51.9
	5-shot	46.5	36.6	17.8	73.1	43.5	60.3	44.9	41.8	81.3	57.1
CoT Prompting†	1-shot	33.8	45.6	33.0	71.2	45.9	31.3	52.6	40.0	77.7	50.4
	5-shot	51.7	43.7	38.1	75.5	52.3	58.5	49.0	43.7	82.5	58.4
AnnoLLM† (NAACL'24)	1-shot	37.4	44.6	16.5	68.3	41.7	45.5	50.2	36.5	74.1	51.6
	5-shot	40.9	43.5	23.9	74.7	45.8	52.1	51.4	40.8	77.7	55.5
PromptAgent (ICLR'24)	1-shot	25.4	45.7	34.1	70.6	44.0	39.3	54.3	35.0	78.2	51.7
	5-shot	42.3	44.2	28.8	72.5	47.0	57.0	49.6	38.7	79.9	56.3
GPO (AAAI'25)	1-shot	26.2	43.0	9.6	68.7	36.9	36.0	48.5	18.7	77.4	45.2
	5-shot	38.7	37.8	15.9	74.8	41.8	55.6	49.5	26.9	80.2	53.1
GDP (ours)	1-shot	**40.1**	**48.4**	**38.3**	**75.0**	**50.5**	**48.5**	**55.5**	**46.8**	**81.6**	**58.1**
	5-shot	**53.5**	**47.1**	**44.7**	**78.8**	**56.0**	**62.1**	**56.5**	**49.5**	**83.4**	**62.9**

Table 1: F1 scores (%) on NER and EE datasets under 5-shot training. Rows correspond to different ICL settings. 1-shot uses the example from the initial prompt and 5-shot includes all training examples. † indicates baselines without few-shot optimization.

Baselines	GPT-3.5-Turbo										GPT-4									
	NCBI		WNUT		Harv.		CASIE		Avg		NCBI		WNUT		Harv.		CASIE		Avg	
	10	20	10	20	10	20	10	20	10	20	10	20	10	20	10	20	10	20	10	20
PromptAgent	45.7	43.6	46.7	45.9	37.0	34.9	72.4	73.5	50.4	49.5	54.4	55.4	50.3	50.7	45.0	32.5	78.1	78.9	56.9	54.4
GPO	34.1	38.5	41.7	42.6	12.5	18.6	69.2	69.4	39.4	42.3	38.8	40.8	47.3	49.7	24.4	26.9	77.6	76.8	47.0	48.6
GDP (ours)	**54.6**	**57.3**	**49.4**	**49.8**	**43.3**	**45.7**	**76.1**	**76.9**	**55.9**	**57.4**	**62.6**	**70.8**	**56.7**	**58.4**	**48.8**	**49.3**	**82.1**	**83.5**	**62.5**	**65.5**

Table 2: F1 scores (%) on NER and EE under 10-shot and 20-shot training. Harv.=HarveyNER for abbreviation.

line segments for each input via RAS and uses them for annotation; (c) Chain-of-Thought Prompting (CoT-P) (Wei et al. 2022): Formats the prompt in a CoT style using guideline steps; (d) AnnoLLM (He et al. 2023): Enhances standard prompting with example-based rationales via CoT; (e) PromptAgent (Wang et al. 2023b) and (f) GPO (Tang et al. 2025): Recent state-of-the-art prompt optimization methods that do not incorporate guideline structure. Initialization and full prompt templates for all baselines are in Appendix G.

Setting To build the RAS $\mathcal{R}$, we use OpenAI's text-embedding-3-large (Brown et al. 2020) as the retriever encoder, and GPT-4o (Achiam et al. 2023) as the query model for interpreting retrieved guideline content. Since most guideline documents are in PDF format, we preprocess and segment them before embedding (details in Appendix C). For inference LLMs, we use GPT-4 as a current state-of-the-art and GPT-3.5-turbo as an average LLM in main experiments. We test open-source LLMs in Appendix E. The tree search runs for 6 iterations, with the top 5 nodes retained as candidates in each round. To avoid human bias, the first prompt P_0 is initialized by $\mathcal{R}$ summarizing the guideline.

Experimental Results

Experimental results are shown in Table 1 for 5-shot training and ICL and Table 2 for 10-shot and 20-shot training. Overall, GDP consistently outperforms all baseline methods across most datasets. Examples of optimized prompts are provided in Appendix I. We observe that prompts produced by GDP are more structured and systematic than those generated by baseline methods. We further analyze GDP's performance through the following comparisons:

GDP adapts better to complex annotation guidelines Performance gains vary across datasets when comparing GDP to the initial summary prompt. This variation correlates with task complexity and guideline richness. For simpler tasks where the guideline contains only basic instructions, summarization followed by RAS retrieval often captures most key requirements. However, for more complex tasks, GDP yields substantial improvements by decomposing the guideline into an executable structure.

Structuring retrieved guidelines enhances reasoning While RAS-based prompting has access to complete guideline instructions, it often includes redundant or overly detailed content. Moreover, LLMs struggle to interpret raw retrieved text without further structuring. In contrast, GDP transforms the retrieved content into a coherent, LLM-friendly workflow that facilitates accurate reasoning.

Dynamic pipeline optimization outperforms static decompositions Although CoT-P attempts to encode step-

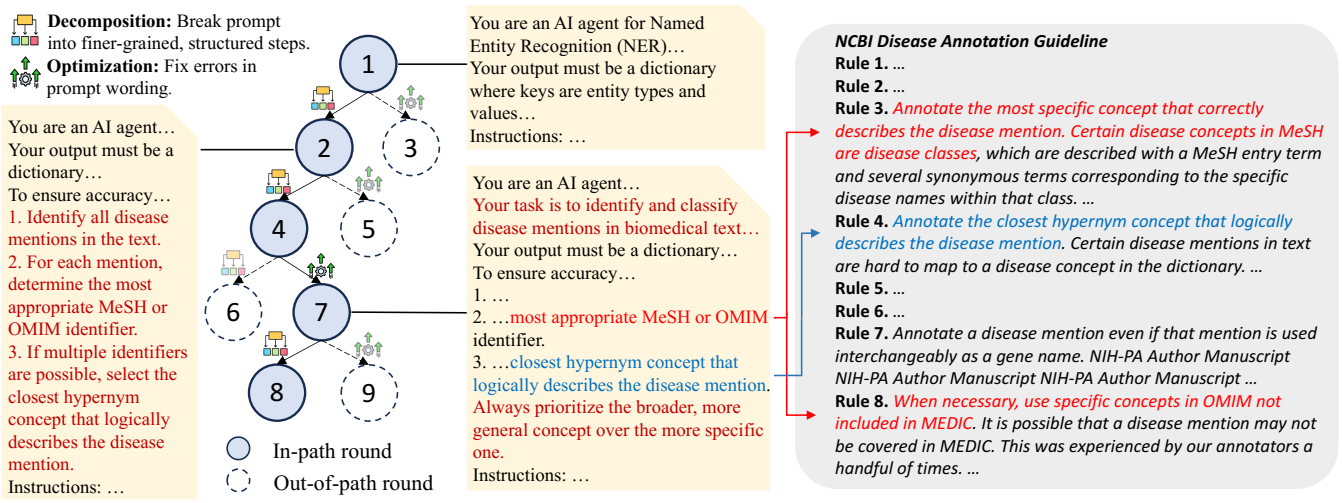

Figure 3: Optimization tree on NCBI-Disease. Dark red indicates updated content. Arrows indicate contents from the guideline.

	NCBI	WNUT	Harv.	CASIE
GDP w/o CO	44.6	55.2	43.7	80.3
GDP w/o PD	42.3	54.8	41.9	80.9
GDP (Ours)	**48.5**	**55.5**	**46.8**	**81.6**

Table 3: Ablations on Extension Strategies. PD means Pipeline Decomposition, CO means Content Optimizations.

	NCBI	WNUT	Harv.	CASIE
PromptAgent w/o $\mathcal{R}$	39.3	54.5	35.0	78.2
PromptAgent w/ $\mathcal{R}$	**41.2**	**56.4**	**41.8**	**80.1**
GPO w/o $\mathcal{R}$	36.0	48.5	18.7	77.4
GPO w/ $\mathcal{R}$	**38.5**	**51.7**	**27.3**	**78.2**

Table 4: Ablations on RAS $\mathcal{R}$, where $\mathcal{R}$ is used for replacing LLM optimizers in baselines.

by-step logic, they do not optimize the pipeline structure itself. In practice, real-world guidelines rarely prescribe a strict execution order, making static designs suboptimal. GDP dynamically adjusts the structure through search, leading to superior performance.

Guideline-driven optimization improves few-shot generalization Existing prompt optimization methods focus on adapting prompts to training examples, but largely ignore guideline structure. As a result, they tend to overfit or bias toward incomplete representations of the task, especially under limited supervision. In contrast, GDP is guided by the full guideline, enabling more robust and generalizable prompt composition.

Analysis on the GDP Algorithm

Child Node Extension To validate the effectiveness of our two extension strategies: pipeline decomposition and content optimization, we conduct an ablation study where each strategy is applied independently. Results are reported in Table 3. Interestingly, we find that content-only optimization still achieves strong performance, suggesting that guideline-driven learning, supported by the RAS $\mathcal{R}$, provides a robust foundation for prompt refinement.

We further illustrate the pipeline optimization process in Figure 3. The figure shows that decomposition is particularly beneficial in early stages of the search, where breaking down the guideline helps incorporate more task-specific detail. In contrast, content optimization proves more effective in later stages, where it fine-tunes the prompt by addressing specific errors using targeted guideline retrieval. This result highlights the complementary nature of the two strategies: decomposition helps expand task structure, while content optimization helps correct fine-grained errors. Additionally, we found GDP can learn prompts with domain specific concepts (red and blue arrows) from the raw guideline while internalize the LLM's understanding of the task (sentence in red) as shown in the 7th prompt.

Effectiveness of $\mathcal{R}$ To evaluate the performance of incorporating the RAS $\mathcal{R}$ for prompt optimization, we replace pure LLM optimizers with $\mathcal{R}$ in training. Results are demonstrated in Table 4. We observe promising improvement after guideline is involved in optimization, which again demonstrate the advantages of guide-drived learning.

Conclusion

We present GDP, a guideline-driven prompt optimization framework that mirrors human-style learning. GDP shifts away from large-scale annotation by synthesizing structured prompts from domain-specific guidelines, using retrieval, decomposition, and iterative refinement. GDP enables LLMs to reason procedurally with interpretable and task-aligned instructions. Unlike implicit data-driven models or flat prompts, GDP yields adaptive, executable representations grounded in expert logic. Experiments across diverse information extraction tasks show that GDP generalizes well under minimal supervision.

Acknowledgments

This work was partially supported by US National Science Foundation IIS-2412195, CCF-2400785, the Cancer Prevention and Research Institute of Texas (CPRIT) award (RP230363), the National Institutes of Health (NIH) R01 award (1R01AI190103-01) and Microsoft Accelerate Foundation Models Research (2024).

References

Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Ashok, D.; and Lipton, Z. C. 2023. Promptner: Prompting for named entity recognition. *arXiv preprint arXiv:2305.15444*.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901.

Chen, P.; Xu, H.; Zhang, C.; and Huang, R. 2022. Crossroads, Buildings and Neighborhoods: A Dataset for Fine-grained Location Recognition. In *NAACL*. Association for Computational Linguistics.

Derczynski, L.; Nichols, E.; Van Erp, M.; and Limsopatham, N. 2017. Results of the WNUT2017 shared task on novel and emerging entity recognition. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, 140–147.

Doğan, R. I.; Leaman, R.; and Lu, Z. 2014. NCBI disease corpus: a resource for disease name recognition and concept normalization. *Journal of biomedical informatics*, 47: 1–10.

Dong, Q.; Li, L.; Dai, D.; Zheng, C.; Ma, J.; Li, R.; Xia, H.; Xu, J.; Wu, Z.; Liu, T.; et al. 2022. A survey on in-context learning. *arXiv preprint arXiv:2301.00234*.

Gunasekar, S.; Zhang, Y.; Aneja, J.; Mendes, C. C. T.; Del Giorno, A.; Gopi, S.; Javaheripi, M.; Kauffmann, P.; de Rosa, G.; Saarikivi, O.; et al. 2023. Textbooks are all you need. *arXiv preprint arXiv:2306.11644*.

He, X.; Lin, Z.; Gong, Y.; Jin, A.; Zhang, H.; Lin, C.; Jiao, J.; Yiu, S. M.; Duan, N.; Chen, W.; et al. 2023. Annollm: Making large language models to be better crowdsourced annotators. *arXiv preprint arXiv:2303.16854*.

James, S.; Konidaris, G.; and Rosman, B. 2017. An analysis of monte carlo tree search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Karpatne, A.; Jia, X.; and Kumar, V. 2024. Knowledge-guided machine learning: Current trends and future prospects. *arXiv preprint arXiv:2403.15989*.

Kumar, B.; Amar, J.; Yang, E.; Li, N.; and Jia, Y. 2024. Selective fine-tuning on llm-labeled data may reduce reliance on human annotation: A case study using schedule-of-event table detection. *arXiv preprint arXiv:2405.06093*.

Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.-t.; Rocktäschel, T.; et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33: 9459–9474.

Min, S.; Lyu, X.; Holtzman, A.; Artetxe, M.; Lewis, M.; Hajishirzi, H.; and Zettlemoyer, L. 2022. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837*.

Petroni, F.; Piktus, A.; Fan, A.; Lewis, P.; Yazdani, M.; De Cao, N.; Thorne, J.; Jernite, Y.; Karpukhin, V.; Maillard, J.; et al. 2020. KILT: a benchmark for knowledge intensive language tasks. *arXiv preprint arXiv:2009.02252*.

Sainz, O.; García-Ferrero, I.; Agerri, R.; de Lacalle, O. L.; Rigau, G.; and Agirre, E. 2023. Gollie: Annotation guidelines improve zero-shot information-extraction. *arXiv preprint arXiv:2310.03668*.

Satyapanich, T.; Ferraro, F.; and Finin, T. 2020. Casie: Extracting cybersecurity event information from text. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 8749–8757.

Tang, X.; Wang, X.; Zhao, W. X.; Lu, S.; Li, Y.; and Wen, J.-R. 2025. Unleashing the potential of large language models as prompt optimizers: Analogical analysis with gradient-based model optimizers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 25264–25272.

Team, G.; Anil, R.; Borgeaud, S.; Alayrac, J.-B.; Yu, J.; Soricut, R.; Schalkwyk, J.; Dai, A. M.; Hauth, A.; Millican, K.; et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Wang, S.; Sun, X.; Li, X.; Ouyang, R.; Wu, F.; Zhang, T.; Li, J.; and Wang, G. 2023a. Gpt-ner: Named entity recognition via large language models. *arXiv preprint arXiv:2304.10428*.

Wang, X.; Li, C.; Wang, Z.; Bai, F.; Luo, H.; Zhang, J.; Jojic, N.; Xing, E. P.; and Hu, Z. 2023b. Promptagent: Strategic planning with language models enables expert-level prompt optimization. *arXiv preprint arXiv:2310.16427*.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q. V.; Zhou, D.; et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35: 24824–24837.

Yao, S.; Yu, D.; Zhao, J.; Shafran, I.; Griffiths, T.; Cao, Y.; and Narasimhan, K. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36: 11809–11822.

Zhang, J.; Xiang, J.; Yu, Z.; Teng, F.; Chen, X.; Chen, J.; Zhuge, M.; Cheng, X.; Hong, S.; Wang, J.; et al. 2024. Aflow: Automating agentic workflow generation. *arXiv preprint arXiv:2410.10762*.

Zhao, S.; Dang, J.; and Grover, A. 2023. Group preference optimization: Few-shot alignment of large language models. *arXiv preprint arXiv:2310.11523*.

Zhu, B.; Niu, Y.; Han, Y.; Wu, Y.; and Zhang, H. 2023. Prompt-aligned gradient for prompt tuning. In *Proceedings of the IEEE/CVF international conference on computer vision*, 15659–15669.