BITEXTION: Sociotechnical Alignment in Automated Optimization Modeling via Textual Bisection for Task Disambiguation

Batu El *
batuel@stanford.edu

Madeleine Udell†
udell@stanford.edu

Abstract

When interacting with automated optimization modeling systems, such as Ahmaditeshnizi et al. [2024], the burden of clearly and precisely describing complex problems falls upon the users, which can be challenging, especially for non-experts, who are the primary user base for these systems. This observation motivates our research to develop a system that can ask clarifying questions to help non-expert users precisely describe complex optimization problems. We introduce Textual Bisection (BITEXTION) a general framework that (i) samples candidate *detailed problem descriptions* conditioned on a *vague problem description*, and (ii) narrows down the search space by posing questions that efficiently resolve ambiguities. Our simulated user study provides experimental evidence showing that BITEXTION improves the performance of Llama3.1-8b when eliciting the objective and constraints of an optimization problem. We find that BITEXTION can enhance the performance of smaller models but may not consistently deliver similar gains when applied to more capable larger models, underscoring the need for model-specific test-time strategies.[3]

1 Introduction

Optimization problems are widespread in the sciences and across industries. Although a large body of academic literature has studied methods to solve optimization problems efficiently, using these state-of-the-art optimization technologies requires one to express the problem in a programming language [Diamond and Boyd, 2016]. Traditionally, this process has relied on human expertise, until recent works have developed systems that convert the natural language descriptions of optimization problems into solver code, which can then be executed to obtain the solutions for optimization problems, bridging the description of an optimization problem to its solution [Ahmaditeshnizi et al., 2024, Astorga et al., 2024, Tang et al., 2024]. These efforts to automate optimization modeling demonstrate significant promise. However, their performance depends heavily on the precise description of complex problems, a task that can be particularly challenging for non-experts, who constitute the primary user base for these systems. This points to a need for developing systems that can interact with users by asking questions to help them precisely describe complex optimization problems.

Research Question

How can we develop a system that can ***efficiently elicit relevant missing information*** from the users by asking clarifying questions?

*Institute for Computational and Mathematical Engineering, Stanford University.
†Management Science and Engineering, Stanford University.
[3]The code for the project may be found at `https://github.com/batu-el/bitextion`.

Preprint.

Throughout the paper, we focus on a simplified version of the problem, where the user is only allowed to give "Yes/No" answers to the questions posed by the system. This simplifies our task to be a version of the *Game of* 20 *Questions*, which has previously been studied to adversarially benchmark LLMs' performance [Richardeau et al., 2024]. In Section 2, we start by formalizing our research question. In Section 3, we propose BITEXTION as a novel approach to efficiently elicit relevant information from the users. We describe our experimental setup in Section 4 before presenting our results in Section 5. Section 6 reviews the related literature. Finally, Section 7 discusses the implications of our findings and proposes potential directions for future work.

2 Formal Setting

We consider a scenario where a user aims to solve an optimization problem, which is fully characterized by a detailed description, $d^\star$. We assume that the user knows all the relevant information about the problem but does not know which details are most relevant. Hence, the user begins the conversations with an initial statement, v, which is a vague expression of a detailed problem description, $d^\star$, but can accurately share information when it is directly requested.[4] Following v, the conversation between the user and the system proceeds in question and answer format, where the system poses a question q_i to which the user responds with $r_i = \mathcal{R}(q_i)$. Throughout the paper, we focus on a simplified setting, where the user is only allowed to give "Yes/No" answers, i.e., $r_i \in \{0, 1\}$ for all i.

We denote the list of k questions that are asked by the system with $\mathbf{q}_k = [q_1, ..., q_k]$, and the associated responses with $\mathbf{r}_k = \mathcal{R}(\mathbf{q}_k) = [r_1, ..., r_k]$. The k^{th} question depends on all the previous questions and the associated responses, as well as the initial vague problem description, i.e.,

$$q_k = \mathcal{Q}(\mathbf{q}_{k-1}, \mathbf{r}_{k-1}, v) = \mathcal{Q}([q_1, ..., q_{k-1}], [r_1, ..., r_{k-1})], v)$$

At step, k, $P_\theta(.|x_{k-1})$ defines a probability distribution over the candidate detailed descriptions conditioned on the conversation history, $x_{i-1} = (\mathbf{q}_{k-1}, \mathbf{r}_{k-1}, v)$. The conversation history includes the initial vague problem description (v), past questions ($\mathbf{q}_{k-1}$), and the corresponding responses ($\mathbf{r}_{k-1}$). From this probability distribution, one can sample candidate detailed descriptions, d_k:

$$d_k \sim P_\theta(.|x_{k-1})$$

for $k = 1, ..., n$. Note that $x_0 = (\mathbf{q}_0, \mathbf{r}_0, v) = (\emptyset, \emptyset, v)$, and $d_1 \sim P_\theta(.|v)$.

We use the loss function $\mathcal{L}(d_k, d^*)$ to evaluate how well the system's prediction of the detailed problem description, d_k, matches the true detailed problem description, $d^\star$. The prediction d_k is sampled from $P_\theta(.|x_{k-1})$. Therefore, the task of predicting $d^\star$ reduces to updating our beliefs about $d^\star$, as encoded by the distribution $P_\theta(. \mid x_{k-1})$, and further simplifies to the formulation of questions q_i that efficiently elicit the most relevant information.

In this setting, we face a tradeoff between the total cost the user incurs while responding to k questions, $C(\mathbf{q}_k)$, and loss, $\mathcal{L}(d_k, d^*)$:

$$\alpha C(\mathbf{q}_k) + (1 - \alpha)\mathcal{L}(d_k, d^*)$$

where $\alpha \in [0, 1]$ is a scalar that trades off loss for cost. A similar trade-off is observed in the preference elicitation setting by Li et al. [2023]. For simplicity, in this paper, we assume that the question budget is fixed at K questions, which allows us to focus only on $\mathcal{L}(d_k, d^*)$, i.e., $\alpha = 0$. Hence, our simplified objective becomes

$$\min_{\mathcal{Q}} \mathcal{L}(d_k, d^*) = \min_{\mathcal{Q}} \mathcal{L}(P_\theta(\mathbf{q}_k, \mathbf{r}_k, v), d^*), \text{ where } q_i = \mathcal{Q}(\mathbf{q}_{i-1}, \mathbf{r}_{i-1}, v) \text{ for } i = 1, ..., k$$

where $k = 1, ..., n$. Note that this formulation defines the loss for each step, k, in the conversation. In Section 3, we present a method for selecting the next question, $\mathcal{Q}$, which we refer to as textual bisection, or BITEXTION.

3 Method

3.1 Bisection and Binary Search (Background)

Traditionally, the bisection method refers to a numerical root-finding algorithm that can be used to identify the root of a continuous function, often a polynomial, $f(x)$, inside the interval $x \in [a, b]$,

[4] While our experimental design incorporates this assumption, due to randomness in language generation, our simulated users may give untruthful responses.

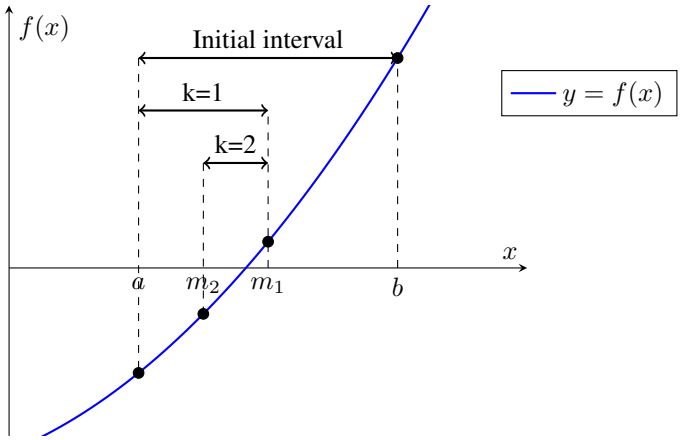

Figure 1: The bisection method for finding a root of a polynomial. In this example, $m_1 = \frac{a+b}{2}$ and $m_2 = \frac{a+m_1}{2}$.

where $f(a)f(b) < 0$, i.e. $f(a)$ and $f(b)$ have opposite signs. The condition $f(a)f(b) < 0$ ensures that there exists an $x \in [a, b]$ such that $f(x) = 0$ by intermediate value theorem. The algorithm identifies the midpoint, $m_1 = \frac{a+b}{2}$, that *bisects* the interval into two equal sub-intervals and evaluates the value of the function f at m_1. If $f(m_1)f(b) < 0$, there exists $x \in [m_1, b]$ such that $f(x) = 0$ by intermediate value theorem. Similarly, if $f(a)f(m_1) < 0$, there exists $x \in [a, m_1]$ such that $f(x) = 0$. We know that either $f(m_1)f(b) < 0$ or $f(a)f(m_1) < 0$ holds, because $f(a)$ and $f(b)$ have opposite signs. Accordingly, we can narrow down our search space to be half of the current size at each iteration of the algorithm and run bisection recursively in the updated search space. Figure 1 illustrates the bisection algorithm.

Guiding Principle

We can efficiently search a domain by dividing the search space into two halves and discarding one half at each iteration of the search algorithm.

The guiding principle of this root-finding algorithm, dividing the search space into two parts and eliminating one half at each iteration, motivates binary search over sorted arrays, a widely used search algorithm with $O(\log(n))$ time complexity.

3.2 BITEXTION in Latent Space (Motivation)

Figure 2 demonstrates a geometric analogy that explains the mechanics of BITEXTION in a 2-dimensional latent space. This space can be thought of as the projection of the embedding space for the detailed problem descriptions onto two principal directions. Each point in this space represents a detailed problem description, d_k^j, which we depict with the blue dots in Figure 2. The red star in the figure represents the true detailed problem description, d^*. At iteration k, a bisecting hyperplane with the normal vector q_k is chosen. Based on the user's response to q_k, the search domain is updated to be the blue shaded region. Consequently, the search domain is narrowed at each iteration, allowing the system to predict d^* with higher precision. Algorithm 1 formalizes this idea in 4 main steps:[5]

1. **Sampling.** Sample candidates $d_k^j \sim P_\theta(.|x = (\mathbf{q}_{k-1}, \mathbf{r}_{k-1}, v)), \quad j \in \{1, 2, \ldots, m\} = \mathcal{C}$
2. **Bisection.** Select q_k such that $q_k^T d_k^j > b \quad$ for $j \in \mathcal{C}_1, \quad q_k^T d_k^j < b \quad$ for $j \in \mathcal{C}_2$ with $\mathcal{C}_1 \cup \mathcal{C}_2 = \mathcal{C}, \quad \mathcal{C}_1 \cap \mathcal{C}_2 = \emptyset, \quad |\mathcal{C}_1| = |\mathcal{C}_2| = \frac{m}{2}$ for some $b \in \mathbb{R}$.
3. **Question.** Pose q_k to the user and record the response $r_k \in \{0, 1\}$.
4. **Domain Update.** Narrow the search domain by conditioning $P_\theta(.|x)$ on the updated context $x = (\mathbf{q}_k, \mathbf{r}_k, v)$.

[5]In section 3.2, we use $d_k^j, q_k, r_k \in \mathbb{R}^d$ as the representations of natural language objects in the latent space.

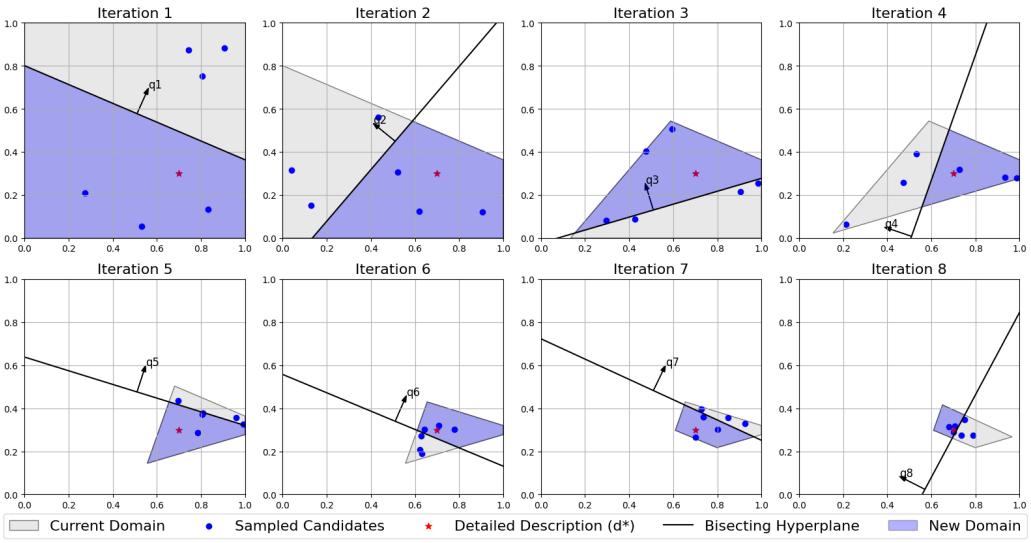

Figure 2: Geometric analogy that explains the mechanics of BITEXTION in a 2-dimensional latent space, which can be thought of as the projection of the embedding space for the detailed problem descriptions onto 2 principal directions ($K = 8, m = 6, d = 2$).

Algorithm 1 BITEXTION in Latent Space

Initialization.
Let v represent the initial vague problem description.
Let $P_\theta(.|x)$ be the probability distribution over candidate descriptions, conditioned on context x.
Set $\mathbf{q}_0 = \emptyset$, $\mathbf{r}_0 = \emptyset$.

for k **in** $[1, \ldots, n]$ **do**
  Step 1. Sampling. Sample m candidate descriptions:

$$d_k^j \sim P_\theta(.|x_{k-1} = (\mathbf{q}_{k-1}, \mathbf{r}_{k-1}, v)), \quad j \in \{1, 2, \ldots, m\} \text{ , where } d_k^j \in \mathbb{R}^d$$

  Step 2. Bisection.
  Identify a balanced partition $(\mathcal{C}_1, \mathcal{C}_2)$ satisfying:

$$\mathcal{C}_1 \cup \mathcal{C}_2 = \mathcal{C}, \quad \mathcal{C}_1 \cap \mathcal{C}_2 = \emptyset, \quad |\mathcal{C}_1| = |\mathcal{C}_2| = \frac{m}{2}.$$

  Find a hyperplane with normal vector q_k and $b \in \mathbb{R}$ such that

$$q_k^T d_k^j > b \quad \text{for } j \in \mathcal{C}_1, \quad q_k^T d_k^j < b \quad \text{for } j \in \mathcal{C}_2.$$

  Step 3. Question. Pose q_k to the user and record the response r_k.

$$r_k = \begin{cases} 1 & \text{if } q_k^\top d^* > b \\ 0 & \text{if } q_k^\top d^* < b \end{cases}$$

  Step 4. Domain Update. Narrow down the search domain by updating the context, x_k.

$$\mathbf{q}_k = \mathbf{q}_{k-1} \cup \{q_k\}, \quad \mathbf{r}_k = \mathbf{r}_{k-1} \cup \{r_k\}, \quad x_k = (\mathbf{q}_k, \mathbf{r}_k, v).$$

end for
Output: $P_\theta(.|x_n)$

In this section, we motivated our method within the latent embedding space. We note that there may be different approaches to convert a latent question, q_k, to a natural language question to be posed to the user, such as training a question generation model conditional on the latent question representations, or, more interestingly, identifying a map between latent question and a direction in

the internal activation space of the model to steer the generation process without additional training [Templeton et al., 2024, Meng et al., 2023]. In our implementation, we take a simpler path, performing each step with operations on natural language objects.

3.3 BITEXTION in Natural Language (Our Method)

3.3.1 Sampling

A language model, P, defines a probability distribution over the completions y conditional on the prefix x. This distribution can be denoted as

$$y \sim P_\theta(.|x)$$

where θ is a combination of model parameters and a system prompt that parameterizes the generation process.[6] This allows us to use P_θ by conditioning on $x = (\mathbf{q}_{k-1}, \mathbf{r}_{k-1}, v)$ to sample candidates d_k^j for our guesses of the detailed problem description from the probability distribution $P_\theta(.|x = (\mathbf{q}_{k-1}, \mathbf{r}_{k-1}, v))$.[7] At each iteration, BITEXTION algorithm samples m candidate detailed descriptions by calling P_θ m times, independently, with the sampling prompt presented in Appendix A.1. The temperature parameter of the language model can be used to adjust the variance of the sampled candidates.[8][9]

3.3.2 Bisection

After generating the samples d_k^j, we identify a balanced partition defined by some choice of $\mathcal{C}_1$ and $\mathcal{C}_2$. To perform this operation on natural language objects, we parameterize a language model with the bisection prompt presented in Appendix A.2.

3.3.3 Question

In this step, we pose the question that our algorithm generated to a user and receive an answer ("Yes" or "No") in return. In our experiments, we use simulated users, which are language model agents who role-play as human users. The details of the simulated user study are described in Section 4.3.

3.3.4 Domain Update

Based on the answer, we update the search domain by adding the question-answer pair in the context before sampling the candidate detailed descriptions d_k at iteration k from $P_\theta(.|x = (\mathbf{q}_{k-1}, \mathbf{r}_{k-1}, v))$. Adding the question-response pairs in the context conditions the generation of candidate detailed descriptions on the recent information attained during the conversation. In the latent space, this step corresponds to narrowing down the search space at each iteration, as demonstrated in Figure 2. In the space of natural language, this corresponds to a shift in the probability distribution defined by the language model, which now allocates less weight to the candidate detailed descriptions that are inconsistent with the recently revealed information.

4 Experimental Setup

4.1 Data Generation

In our experiments, we use 3 datasets: NLP4LP from Ahmaditeshnizi et al. [2024], NL4OPT from Ramamonjison et al. [2022], and ComplexOR from Xiao et al. [2024]. We take the detailed problem descriptions, d^*, from these datasets, and generate vague problem descriptions, v, based on d^*. Language models tend to mirror the format of the prompt in their responses; therefore, we use a

[6] The system prompt is a high-level instruction in natural language that guides generation.

[7] In Section 3.3, we use d_k^j, q_k, r_k to refer to the natural language objects.

[8] We experiment with $t = 0.8$; however, quantifying the impact of the variance of samples on the performance of BITEXTION is outside the scope of our experiments.

[9] We mention in passing that mode collapse [janus, 2023, Dohmatob et al., 2024], a phenomenon commonly observed in generative models, can reduce the diversity of the samples, undermining the performance of BITEXTION.

vague, one-sentence prompt to generate vague problem descriptions (see Prompt Engineering Best Practices). We prompt GPT-4o [OpenAI et al., 2024] to generate synthetic vague problem descriptions with the data generation prompt presented in Appendix A.3. All three of our datasets contain (i) a detailed problem description, (ii) a vague problem description, which we synthetically generate based on the detailed description, (iii) target parameters, (iv) target objective, and (v) target constraints. The target fields are generated by humans based on the detailed descriptions of the problems and were available in the versions of the dataset from AhmadiTeshnizi et al. [2024].

Dataset Name	**Num. Constraints**	**Detailed Desc. Len.**	**Vague Desc. Len.**
complexor ($n = 20$)	3.80 ± 1.44	103.00 ± 26.15	16.95 ± 4.97
nl4opt ($n = 285$)	5.69 ± 2.01	100.37 ± 18.14	20.36 ± 5.58
nlp4lp ($n = 65$)	3.79 ± 2.10	67.21 ± 29.81	17.35 ± 4.90

Table 1: Dataset Statistics. Standard deviations are presented on the right side of the $\pm$ sign.

Table 1 presents the number of examples (n), the number of constraints per problem (Num. Constraints), and the number of words in the detailed and vague problem descriptions (Vague Desc. Len. and Detailed Desc. Len.). The datasets are linked in our GitHub repository. Figure 3 shows a comparison of the vague description lengths and the detailed description lengths in these datasets.

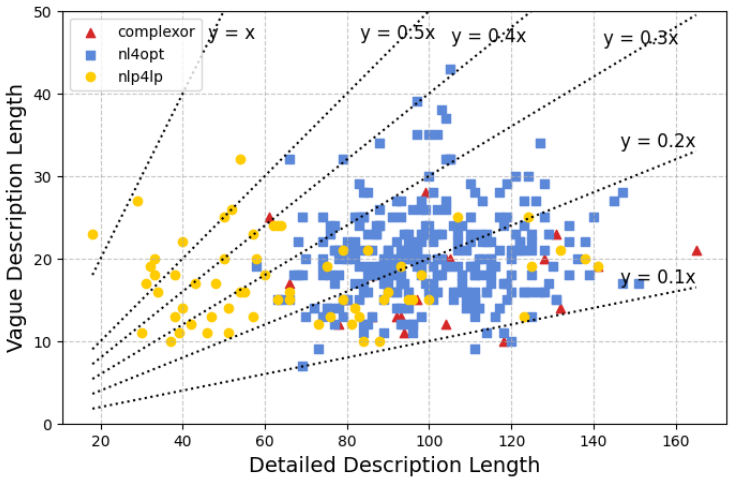

Figure 3: Vague Description (v) and Detailed Description (d^*) Length Comparison for NLP4LP, ComplexOR, and NL4OPT datasets.

4.2 Baseline

For our baseline approach, instead of using BITEXTION, we prompt the language model to come up with a question using the baseline prompt presented in Appendix A.4.

4.3 Simulated User Study

To evaluate the performance of our approach compared to the baseline, we conduct a simulated user study. In this study, we present questions generated by BITEXTION and the baseline to a simulated user, who responds with either "Yes" or "No." While a real user study would be the gold standard to evaluate our method, hiring real humans is costly. Therefore, we rely on a simulated user study, which, despite certain shortcomings, such as inconsistencies in responses, is a cost-effective proxy for human evaluation [Huang et al., 2024]. We parameterize a language model with the user prompt (presented in Appendix A.5) to roleplay as a user. In all experiments, we use GPT-4o [OpenAI et al., 2024] to roleplay as the user. Only when formulating questions, we employ two different models:

GPT-4o and Llama3.1-8b [OpenAI et al., 2024, Grattafiori et al., 2024].[10] We set the number of samples $m = 6$ in all runs (see Section 3.3.1) and set temperature $t = 0.8$ while sampling.[11] In all our experiments, we generated 3 questions, which corresponds to $K = 4$ conversation steps.

4.4 Evaluation

To evaluate our system, we measure the *alignment* between the user's intent and the system's knowledge of that intent. We sample a detailed description, d_k at each step in the conversation using the sampling prompt from Section 3.3.1. Then, we compare the objective and constraints from this sampled detailed description d_k with the target objective and constraints of d^*. The target objective captures the user's desired outcome, and the target constraints define the permissible conditions that guide how this objective can be pursued. We compare the objective and constraints with 2 independent calls to a language model that is instructed to perform the scoring task with the scoring prompt presented in Appendix A.6. The scores are assigned using GPT-4o [OpenAI et al., 2024] for all experiments.

Finally, we compare the average scores for BITEXTION and the baseline, where the average is taken across examples in each dataset. This value can be interpreted as the inverse of the loss function $\mathcal{L}(d_k, d^*)$, which we described in Section 2. For completeness, we also report the parameter similarity scores in Appendices B and C. In the following section, we present our results.

5 Results

Table 2 demonstrates the results from our experiments across 3 datasets and 2 models. We evaluate the performance of BITEXTION against the baseline using a similarity score, which quantifies the similarity between a detailed description d_k, generated conditioned on the information available at step k, and the true detailed description d^*, as described in Section 4.4.

			NLP4LP ($n = 65$)			**ComplexOR** ($n = 20$)			**NL4OPT** ($n = 70$)		
			$k = 2$	$k = 3$	$k = 4$	$k = 2$	$k = 3$	$k = 4$	$k = 2$	$k = 3$	$k = 4$
Constraint	**GPT-4o**	**B**	0.94	**0.98**	0.81	**1.40**	**1.15**	**1.00**	1.03	**1.14**	**1.13**
		BiTxt	**1.02**	0.84	**0.92**	0.70	0.75	0.65	**1.27**	1.10	0.94
	Llama3.1-8b	**B**	**1.03**	0.85	0.79	0.90	0.80	0.75	1.06	0.94	0.99
		BiTxt	1.00	**1.11**	**1.00**	**1.05**	**1.50**	**1.05**	**1.51**	**1.23**	**1.06**
Objective	**GPT-4o**	**B**	**3.21**	**3.11**	**3.18**	3.40	**3.60**	**3.50**	**2.73**	**2.71**	**2.61**
		BiTxt	3.02	3.10	3.13	3.40	3.30	3.35	2.50	2.56	2.40
	Llama3.1-8b	**B**	**3.27**	3.34	3.05	3.05	3.45	3.35	**2.43**	2.34	2.37
		BiTxt	3.23	3.34	**3.42**	**3.25**	**3.65**	**3.65**	2.30	**2.66**	**2.80**

Table 2: Similarity Scores Averaged Across Examples. B represents the baseline score and BiTxt represents the score for BITEXTION algorithm. Higher similarity to the targets indicates better performance of the method. To mark the cases where BITEXTION performs better compared to the baseline, we use **green highlight**. Similarly, **red highlight** marks the cases where the baseline outperforms BITEXTION. We exclude $k = 1$ from this Table (but include it in Table 3 in Appendix B), since no questions have been asked and no reponses have been received when we sample $d_1 \sim P_\theta(.|x_0 = (\mathbf{q}_0, \mathbf{r}_0, v) = (\emptyset, \emptyset, v))$ in the first iteration. For NL4OPT, we use a randomly selected subset of 70 examples from the 285 examples in the dataset to reduce computational cost of the experiments.

Validation for Evaluation Setup: We note that, on average, GPT-4o baseline achieves higher scores compared to Llama3.1-8b baseline, which is expected because better GPT-4o is a larger and more capable model compared to Llama3.1-8b. This serves as a sanity check for our evaluation setup.

[10] We access GPT-4o via the OpenAI API and Llama3.1-8b via the Together AI API.

[11] This is slightly higher than the default value ($t = 0.7$) and aims to increase the diversity of the samples.

BITEXTION outperforms Llama3.1-8b Baseline: We observe BITEXTION improves the performance of Llama3.1-8b when eliciting the constraints and the objective of the optimization problems from the users. Notably, in constraint elicitation, BITEXTION **outperforms the baseline in 8 out of 9 cases**, while performing marginally worse in one case (see rows 3 and 4 from Table 2). We observe a similar but weaker trend in the objective similarity metrics, with BITEXTION with Llama3.1-8b outperforming Llama3.1-8b baseline in 6 out of 9 cases (see rows 7 and 8 from Table 2).

GPT-4o baseline outperforms BITEXTION : In contrast, the baseline of simply delegating the task of asking questions to GPT-4o outperforms BITEXTION with GPT-4o in 6 out of 9 cases (see rows 1 and 2 from Table 2). Similarly, GPT-4o baseline outperforms BITEXTION with GPT-4o in 8 out of 9 cases (see rows 5 and 6 from Table 2). This highlights that our approach can enhance the performance of smaller models but may not consistently deliver similar gains when applied to larger models.

Comparison Across Vague and Detailed Description Length Ratios: Next, we investigate whether the performance of BITEXTION depends on the ratio between the length of the vague description and the detailed description. Intuitively, one may expect that when the vague description, v, reveals only very little information about the true detailed description, d^*, there is more information to be elicited from the user. We use the ratio between lengths (in terms of word count) of v and d^* as a proxy for the information discrepancy between v and d^*. Consistent with our expectations, across all three datasets, we observe that as this ratio moves from a higher percentage range toward a lower percentage range, the similarity scores decrease on average. In Appendix B, Table 4 presents the average scores for subgroups with different vague to detailed description length ratios. Appendix C includes the corresponding plots. Our regression analysis, presented in Figure 7, demonstrates that there is no consistent trend between the improvement over the baseline similarity score and the ratio between the lengths of the vague and detailed descriptions.

6 Related Work

Automated Optimization Modeling: In recent years, Large Language Models (LLMs) have demonstrated remarkable improvements in their quantitative reasoning abilities [Lightman et al., 2023, OpenAI, 2024, Yang et al., 2024a, Lei et al., 2024, Ramamonjison et al., 2023]. In this paper, we focused on optimization problems, which differ from other classes of mathematical problems because quantitative reasoning often does not suffice to solve real-world optimization problems: Obtaining the solution requires leveraging iterative algorithms, such as the simplex algorithm [Dantzig, 1963], interior point methods [Gondzio, 2012], or gradient descent. These algorithms require step by step computations with mathematical precision that involve tracking intermediate states and performing iterative updates deterministically. Hence, formal programs are an effective way to express their computational recipes. While following the steps of an iterative algorithm in natural language chains of thought may be a feasible approach, as demonstrated in Yang et al. [2024b], doing thousands of iterations in natural language would be terribly inefficient compared to using specialized software that can efficiently handle large-scale optimization problems. Therefore, solving an optimization problem efficiently often involves two steps: (1) converting problem description to code and (2) executing the code to obtain the solution. Previous works on automating optimization modeling [Ramamonjison et al, 2022, Tang et al., 2024, AhmadiTeshnizi et al., 2023, Anonymous, 2024, Astorga et al., 2024, AhmadiTeshnizi et al., 2024] have built systems that map problem descriptions to solver code, leveraging libraries such as Gurobi [Gurobi Optimization, LLC, 2023] and cvxpy [Diamond and Boyd, 2016] that contain efficient implementations of optimization algorithms, thereby reducing the reliance on human expertise, which has been the main bottleneck in solving optimization problems.

Information Elicitation: While the literature on automated optimization modeling has studied mapping problem descriptions to solver code, relatively less attention has been paid to eliciting problem descriptions from users. However, several recent studies have addressed this question from different perspectives. Notably, Tamkin et al. [2022] have studied how language models respond to task ambiguities, motivated by their observation that, unlike benchmarks, real-world tasks are often underspecified. Li et al. [2023] and Handa et al. [2024] have studied eliciting human preferences using language models. Similarly, Andukuri et al. [2024] explored teaching language models to ask better clarifying questions for preference elicitation. Other works have developed information-seeking systems for human-AI collaboration in decision-oriented dialogues [Lawless et al., 2024, Lin et al.,

2024, Chen et al., 2024]. Finally, in relation to our formulation of BITEXTION in the latent space, the literature on contextual search has explored multidimensional generalizations of binary search [Krishnamurthy et al., 2022, Leme and Schneider, 2018, Chen et al., 2022].

7 Discussion

In this paper, we introduced a novel method, BITEXTION, that draws inspiration from the traditional bisection algorithm to improve language models' ability to elicit information from users by efficiently resolving ambiguities in automated optimization modeling. Our simulated user study demonstrated that BITEXTION enhances the performance of Llama3.1-8b in eliciting the constraints and objectives of optimization problems. We observed that while BITEXTION can improve the performance of smaller models, it does not yield improvements for larger models. Our findings underscore the need to develop model-specific strategies to improve the model's performance at test time.

Future work can implement BITEXTION in the latent space by exploring different approaches to convert a latent question, q_k, to a natural language question. These approaches can include training a question generation model conditional on the latent question representations, or, more interestingly, identifying a map between the latent question and a direction in the internal activation space of the model to steer the generation process without additional training [Templeton et al., 2024, Meng et al., 2023]. Our method can be improved by allowing the user to provide open-ended responses and training the system to adaptively choose the number of questions to be asked. Our evaluation can be extended by using diverse user personas in simulated experiments and comparing BITEXTION against the baseline using win ratios in a game-play setting.

We developed BITEXTION targeting automated optimization modeling tasks, but our framework is general and can also be applied to other conversational information elicitation settings. In particular, this paper highlights the connection between information elicitation and alignment, noting that aligning an AI system fundamentally requires understanding the user's intentions. However, users may struggle to fully specify the desired behavior of a model, particularly when addressing edge cases, resulting in misaligned behaviors, such as reward hacking [Pan et al., 2024]. Consequently, aligning AI systems is closely related to the task of teaching AI systems to ask clarifying questions to resolve ambiguities in the user's intentions, which can be framed as a set of objectives and constraints that the user may not be able to precisely articulate but can express through answering questions.

References

Ali AhmadiTeshnizi, Wenzhi Gao, and Madeleine Udell. Optimus: Optimization modeling using mip solvers and large language models, 2023. URL https://arxiv.org/abs/2310.06116.

Ali AhmadiTeshnizi, Wenzhi Gao, Herman Brunborg, Shayan Talaei, and Madeleine Udell. Optimus-0.3: Using large language models to model and solve optimization problems at scale, 2024. URL https://arxiv.org/abs/2407.19633.

Ali Ahmaditeshnizi, Wenzhi Gao, and Madeleine Udell. OptiMUS: Scalable optimization modeling with (MI)LP solvers and large language models. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 577–596. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/ahmaditeshnizi24a.html.

Chinmaya Andukuri, Jan-Philipp Fränken, Tobias Gerstenberg, and Noah D. Goodman. Star-gate: Teaching language models to ask clarifying questions, 2024. URL https://arxiv.org/abs/2403.19154.

Anonymous. Optibench: Benchmarking large language models in optimization modeling with equivalence-detection evaluation. In *Submitted to The Thirteenth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=KD9F5Ap878. under review.

Nicolás Astorga, Tennison Liu, Yuanzhang Xiao, and Mihaela van der Schaar. Autoformulation of mathematical optimization models using llms, 2024. URL https://arxiv.org/abs/2411.01679.

Sanxing Chen, Sam Wiseman, and Bhuwan Dhingra. Chatshop: Interactive information seeking with language agents, 2024. URL https://arxiv.org/abs/2404.09911.

Xi Chen, Quanquan Liu, and Yining Wang. Active learning for contextual search with binary feedbacks, 2022. URL https://arxiv.org/abs/2110.01072.

George B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, Princeton, NJ, 1963. ISBN 978-0-691-08000-5.

Steven Diamond and Stephen Boyd. Cvxpy: A python-embedded modeling language for convex optimization, 2016. URL https://arxiv.org/abs/1603.00943.

Elvis Dohmatob, Yunzhen Feng, and Julia Kempe. Model collapse demystified: The case of regression, 2024. URL https://arxiv.org/abs/2402.07712.

Jacek Gondzio. Interior point methods 25 years later. *European Journal of Operational Research*, 218(3):587–601, 2012. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2011.09.017. URL https://www.sciencedirect.com/science/article/pii/S0377221711008204.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie

Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani,

Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual*, 2023. URL https://www.gurobi.com. Version 12.0.

Kunal Handa, Yarin Gal, Ellie Pavlick, Noah Goodman, Jacob Andreas, Alex Tamkin, and Belinda Z. Li. Bayesian preference elicitation with language models, 2024. URL https://arxiv.org/abs/2403.05534.

Yue Huang, Zhengqing Yuan, Yujun Zhou, Kehan Guo, Xiangqi Wang, Haomin Zhuang, Weixiang Sun, Lichao Sun, Jindong Wang, Yanfang Ye, and Xiangliang Zhang. Social science meets llms: How reliable are large language models in social simulations?, 2024. URL https://arxiv.org/abs/2410.23426.

janus. Mysteries of mode collapse, 2023. URL https://www.lesswrong.com/posts/t9svvNPNmFf5Qa3TA/mysteries-of-mode-collapse. Accessed: 2024-11-25.

Akshay Krishnamurthy, Thodoris Lykouris, Chara Podimata, and Robert Schapire. Contextual search in the presence of adversarial corruptions, 2022. URL https://arxiv.org/abs/2002.11650.

Connor Lawless, Jakob Schoeffer, Lindy Le, Kael Rowan, Shilad Sen, Cristina St. Hill, Jina Suh, and Bahareh Sarrafzadeh. "i want it that way": Enabling interactive decision support using large language models and constraint programming, 2024. URL https://arxiv.org/abs/2312.06908.

Bin Lei, Yi Zhang, Shan Zuo, Ali Payani, and Caiwen Ding. Macm: Utilizing a multi-agent system for condition mining in solving complex mathematical problems, 2024. URL https://arxiv.org/abs/2404.04735.

Renato Paes Leme and Jon Schneider. Contextual search via intrinsic volumes, 2018. URL https://arxiv.org/abs/1804.03195.

Belinda Z. Li, Alex Tamkin, Noah Goodman, and Jacob Andreas. Eliciting human preferences with language models, 2023. URL https://arxiv.org/abs/2310.11589.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023. URL https://arxiv.org/abs/2305.20050.

Jessy Lin, Nicholas Tomlin, Jacob Andreas, and Jason Eisner. Decision-oriented dialogue for human-ai collaboration, 2024. URL https://arxiv.org/abs/2305.20076.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt, 2023. URL https://arxiv.org/abs/2202.05262.

OpenAI. Openai o1 system card, 2024. Available at https://openai.com/index/openai-o1-system-card/.

OpenAI, :, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Mądry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, Alex Nichol, Alex Paino, Alex Renzin, Alex Tachard Passos, Alexander Kirillov, Alexi Christakis, Alexis Conneau, Ali Kamali, Allan Jabri, Allison Moyer, Allison Tam, Amadou Crookes, Amin Tootoochian, Amin Tootoonchian, Ananya Kumar, Andrea Vallone, Andrej Karpathy, Andrew Braunstein, Andrew Cann, Andrew Codispoti, Andrew Galu, Andrew Kondrich, Andrew Tulloch, Andrey Mishchenko, Angela Baek, Angela Jiang, Antoine Pelisse, Antonia Woodford, Anuj Gosalia, Arka Dhar, Ashley Pantuliano, Avi Nayak, Avital Oliver, Barret Zoph, Behrooz Ghorbani, Ben Leimberger, Ben Rossen, Ben Sokolowsky, Ben Wang, Benjamin Zweig, Beth Hoover, Blake Samic, Bob McGrew, Bobby Spero, Bogo Giertler, Bowen Cheng, Brad Lightcap, Brandon Walkin, Brendan Quinn, Brian Guarraci, Brian Hsu, Bright Kellogg, Brydon Eastman, Camillo Lugaresi, Carroll Wainwright, Cary Bassin, Cary Hudson, Casey Chu, Chad Nelson, Chak Li, Chan Jun Shern, Channing Conger, Charlotte Barette, Chelsea Voss, Chen Ding, Cheng Lu, Chong Zhang, Chris Beaumont, Chris Hallacy, Chris Koch, Christian Gibson, Christina Kim, Christine Choi, Christine McLeavey, Christopher Hesse, Claudia Fischer, Clemens Winter, Coley Czarnecki, Colin Jarvis, Colin Wei, Constantin Koumouzelis, Dane Sherburn, Daniel Kappler, Daniel Levin, Daniel Levy, David Carr, David Farhi, David Mely, David Robinson, David Sasaki, Denny Jin, Dev Valladares, Dimitris Tsipras, Doug Li, Duc Phong Nguyen, Duncan Findlay, Edede Oiwoh, Edmund Wong, Ehsan Asdar, Elizabeth Proehl, Elizabeth Yang, Eric Antonow, Eric Kramer, Eric Peterson, Eric Sigler, Eric Wallace, Eugene Brevdo, Evan Mays, Farzad Khorasani, Felipe Petroski Such, Filippo Raso, Francis Zhang, Fred von Lohmann, Freddie Sulit, Gabriel Goh, Gene Oden, Geoff Salmon, Giulio Starace, Greg Brockman, Hadi Salman, Haiming Bao, Haitang Hu, Hannah Wong, Haoyu Wang, Heather Schmidt, Heather Whitney, Heewoo Jun, Hendrik Kirchner, Henrique Ponde de Oliveira Pinto, Hongyu Ren, Huiwen Chang, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian O'Connell, Ian Osband, Ian Silber, Ian Sohl, Ibrahim Okuyucu, Ikai Lan, Ilya Kostrikov, Ilya Sutskever, Ingmar Kanitscheider, Ishaan Gulrajani, Jacob Coxon, Jacob Menick, Jakub Pachocki, James Aung, James Betker, James Crooks, James Lennon, Jamie Kiros, Jan Leike, Jane Park, Jason Kwon, Jason Phang, Jason Teplitz, Jason Wei, Jason Wolfe, Jay Chen, Jeff Harris, Jenia Varavva, Jessica Gan Lee, Jessica Shieh, Ji Lin, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joanne Jang, Joaquin Quinonero Candela, Joe Beutler, Joe Landers, Joel Parish, Johannes Heidecke, John Schulman, Jonathan Lachman, Jonathan McKay, Jonathan Uesato, Jonathan Ward, Jong Wook Kim, Joost Huizinga, Jordan Sitkin, Jos Kraaijeveld, Josh Gross, Josh Kaplan, Josh Snyder, Joshua Achiam, Joy Jiao, Joyce Lee, Juntang Zhuang, Justyn Harriman, Kai Fricke, Kai Hayashi, Karan Singhal, Katy Shi, Kavin Karthik, Kayla Wood, Kendra Rimbach, Kenny Hsu, Kenny Nguyen, Keren Gu-Lemberg, Kevin Button, Kevin Liu, Kiel Howe, Krithika Muthukumar, Kyle Luther, Lama Ahmad, Larry Kai, Lauren Itow, Lauren Workman, Leher Pathak, Leo Chen, Li Jing, Lia Guy, Liam Fedus, Liang Zhou, Lien Mamitsuka, Lilian Weng, Lindsay McCallum, Lindsey Held, Long Ouyang, Louis Feuvrier, Lu Zhang, Lukas Kondraciuk, Lukasz Kaiser, Luke Hewitt, Luke Metz, Lyric Doshi, Mada Aflak, Maddie Simens, Madelaine Boyd, Madeleine Thompson, Marat Dukhan, Mark Chen, Mark Gray, Mark Hudnall, Marvin Zhang, Marwan Aljubeh, Mateusz Litwin, Matthew Zeng, Max Johnson, Maya Shetty, Mayank Gupta, Meghan Shah, Mehmet Yatbaz, Meng Jia Yang, Mengchao Zhong, Mia Glaese, Mianna Chen, Michael Janner, Michael Lampe, Michael Petrov, Michael Wu, Michele Wang, Michelle Fradin, Michelle Pokrass, Miguel Castro, Miguel Oom Temudo de Castro, Mikhail Pavlov, Miles Brundage, Miles Wang, Minal Khan, Mira Murati, Mo Bavarian, Molly Lin, Murat Yesildal, Nacho Soto, Natalia Gimelshein, Natalie Cone, Natalie Staudacher, Natalie Summers, Natan LaFontaine, Neil Chowdhury, Nick Ryder, Nick Stathas, Nick Turley, Nik Tezak, Niko Felix, Nithanth Kudige, Nitish Keskar, Noah Deutsch, Noel Bundick, Nora Puckett, Ofir Nachum, Ola Okelola, Oleg Boiko, Oleg Murk, Oliver Jaffe, Olivia Watkins, Olivier Godement, Owen Campbell-Moore, Patrick Chao, Paul McMillan, Pavel Belov, Peng Su, Peter Bak, Peter Bakkum, Peter Deng, Peter Dolan, Peter Hoeschele, Peter Welinder, Phil Tillet, Philip Pronin, Philippe Tillet, Prafulla Dhariwal, Qiming Yuan, Rachel Dias, Rachel Lim, Rahul Arora, Rajan Troll, Randall Lin, Rapha Gontijo Lopes, Raul Puri, Reah Miyara, Reimar Leike, Renaud Gaubert, Reza Zamani, Ricky Wang, Rob Donnelly, Rob Honsby, Rocky Smith, Rohan Sahai, Rohit Ramchandani, Romain Huet, Rory Carmichael, Rowan Zellers, Roy Chen, Ruby Chen, Ruslan Nigmatullin, Ryan Cheu, Saachi

Jain, Sam Altman, Sam Schoenholz, Sam Toizer, Samuel Miserendino, Sandhini Agarwal, Sara Culver, Scott Ethersmith, Scott Gray, Sean Grove, Sean Metzger, Shamez Hermani, Shantanu Jain, Shengjia Zhao, Sherwin Wu, Shino Jomoto, Shirong Wu, Shuaiqi, Xia, Sonia Phene, Spencer Papay, Srinivas Narayanan, Steve Coffey, Steve Lee, Stewart Hall, Suchir Balaji, Tal Broda, Tal Stramer, Tao Xu, Tarun Gogineni, Taya Christianson, Ted Sanders, Tejal Patwardhan, Thomas Cunninghman, Thomas Degry, Thomas Dimson, Thomas Raoux, Thomas Shadwell, Tianhao Zheng, Todd Underwood, Todor Markov, Toki Sherbakov, Tom Rubin, Tom Stasi, Tomer Kaftan, Tristan Heywood, Troy Peterson, Tyce Walters, Tyna Eloundou, Valerie Qi, Veit Moeller, Vinnie Monaco, Vishal Kuo, Vlad Fomenko, Wayne Chang, Weiyi Zheng, Wenda Zhou, Wesam Manassra, Will Sheu, Wojciech Zaremba, Yash Patil, Yilei Qian, Yongjik Kim, Youlong Cheng, Yu Zhang, Yuchen He, Yuchen Zhang, Yujia Jin, Yunxing Dai, and Yury Malkov. Gpt-4o system card, 2024. URL https://arxiv.org/abs/2410.21276.

Alexander Pan, Erik Jones, Meena Jagadeesan, and Jacob Steinhardt. Feedback loops with language models drive in-context reward hacking, 2024. URL https://arxiv.org/abs/2402.06627.

Rindranirina Ramamonjison, Timothy Yu, Raymond Li, Haley Li, Giuseppe Carenini, Bissan Ghaddar, Shiqi He, Mahdi Mostajabdaveh, Amin Banitalebi-Dehkordi, Zirui Zhou, and Yong Zhang. Nl4opt competition: Formulating optimization problems based on their natural language descriptions. In Marco Ciccone, Gustavo Stolovitzky, and Jacob Albrecht, editors, *Proceedings of the NeurIPS 2022 Competitions Track*, volume 220 of *Proceedings of Machine Learning Research*, pages 189–203. PMLR, 28 Nov–09 Dec 2022. URL https://proceedings.mlr.press/v220/ramamonjison23a.html.

Rindranirina Ramamonjison, Timothy T. Yu, Raymond Li, Haley Li, Giuseppe Carenini, Bissan Ghaddar, Shiqi He, Mahdi Mostajabdaveh, Amin Banitalebi-Dehkordi, Zirui Zhou, and Yong Zhang. Nl4opt competition: Formulating optimization problems based on their natural language descriptions, 2023. URL https://arxiv.org/abs/2303.08233.

. Ramamonjison et al. Augmenting operations research with auto-formulation of optimization models from problem descriptions. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 29–62, Abu Dhabi, UAE, December 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.emnlp-industry.4.

Gurvan Richardeau, Erwan Le Merrer, Camilla Penzo, and Gilles Tredan. The 20 questions game to distinguish large language models, 2024. URL https://arxiv.org/abs/2409.10338.

Alex Tamkin, Kunal Handa, Avash Shrestha, and Noah Goodman. Task ambiguity in humans and language models, 2022. URL https://arxiv.org/abs/2212.10711.

Zhengyang Tang, Chenyu Huang, Xin Zheng, Shixi Hu, Zizhuo Wang, Dongdong Ge, and Benyou Wang. Orlm: Training large language models for optimization modeling, 2024. URL https://arxiv.org/abs/2405.17743.

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L Turner, Callum McDougall, Monte MacDiarmid, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*, 2024. URL https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html.

Ziyang Xiao, Dongxiang Zhang, Yangjun Wu, Lilin Xu, Yuan Jessica Wang, Xiongwei Han, Xiaojin Fu, Tao Zhong, Jia Zeng, Mingli Song, and Gang Chen. Chain-of-experts: When LLMs meet complex operations research problems. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=HobyL1B9CZ.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement, 2024a. URL https://arxiv.org/abs/2409.12122.

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

A Prompts

A.1 Sampling Prompt

Sampling Prompt

You are an expert in solving optimization problems. Your task is to predict a detailed optimization problem based on information from a partially disclosed problem and the accompanying conversation. Analyze the given context carefully and provide a comprehensive and plausible problem description that aligns with the provided details.

A.2 Bisection Prompt

Bisection Prompt

You are an expert in solving optimization problems. Your goal is to uncover key details about the parameters, constraints, and objectives of a partially disclosed optimization problem by asking precise, targeted questions. The user has relevant information but may not know which details are critical. Formulate one concise yes/no question designed to divide the candidate problem descriptions into two distinct groups, where the answer is 'yes' for half of the descriptions and 'no' for the other half.

A.3 Data Generation Prompt

Data Generation Prompt

Give a vague single-sentence description of the problem.

A.4 Baseline Prompt

Baseline Prompt

You are an expert in solving optimization problems. Your goal is to uncover key information about the parameters, constraints, and objective of an undisclosed optimization problem by asking insightful, targeted questions. The other agent has this information but does not know which details are relevant. You may ask questions one by one to extract yes/no responses from the other agent.

A.5 User Prompt

User Prompt

You are a user with the following optimization problem: d^* You will answer questions based on what is explicitly asked, providing accurate 'Yes' or 'No' answers.

A.6 Scoring Prompt

Scoring Prompt

Evaluate the similarity between the predicted **parameters/objective/constraints** and the target **parameters/objective/constraints** . Ignore the structure, wording, and specific phrasing. Focus only on the core meaning. Score the similarity on a scale of 0 to 5 based on how well the predicted objective and target objective match:
0: No similarity, **1:** Low similarity, **2:** Partial similarity, **3:** Moderate similarity,
4: High similarity, **5:** Near-exact or exact match.

B Supplementary Tables

NLP4LP ($n = 65$)												
	Constraint Similarity				**Objective Similarity**				**Parameter Similarity**			
	GPT-4o		**Llama3.1-8b**		**GPT-4o**		**Llama3.1-8b**		**GPT-4o**		**Llama3.1-8b**	
k	**B**	**BiTxt**	**B**	**BiTxt**	**B**	**BiTxt**	**B**	**BiTxt**	**B**	**BiTxt**	**B**	**BiTxt**
1	1.11	1.18	1.18	1.19	3.34	3.45	3.19	3.44	1.23	1.27	1.16	1.13
2	0.94	**1.02**	**1.03**	1.00	**3.21**	3.02	**3.27**	3.23	**1.15**	1.08	**1.06**	1.02
3	**0.98**	0.84	0.85	**1.11**	**3.11**	3.10	3.34	3.34	**1.15**	0.84	1.03	**1.06**
4	0.81	**0.92**	0.79	**1.00**	**3.18**	3.13	3.05	**3.42**	**1.13**	0.85	**0.92**	0.82
ComplexOR ($n = 20$)												
	Constraint Similarity				**Objective Similarity**				**Parameter Similarity**			
	GPT-4o		**Llama3.1-8b**		**GPT-4o**		**Llama3.1-8b**		**GPT-4o**		**Llama3.1-8b**	
k	**B**	**BiTxt**	**B**	**BiTxt**	**B**	**BiTxt**	**B**	**BiTxt**	**B**	**BiTxt**	**B**	**BiTxt**
1	1.35	1.30	1.25	1.25	3.65	3.80	3.65	3.40	1.55	1.30	1.30	1.45
2	**1.40**	0.70	0.90	**1.05**	3.40	3.40	3.05	**3.25**	**1.20**	0.85	**1.10**	0.95
3	**1.15**	0.75	0.80	**1.50**	**3.60**	3.30	3.45	**3.65**	**0.85**	0.75	**1.15**	1.10
4	**1.00**	0.65	0.75	**1.05**	**3.50**	3.35	3.35	**3.65**	**1.20**	0.65	**1.00**	0.95
NL4OPT ($n = 70$)												
	Constraint Similarity				**Objective Similarity**				**Parameter Similarity**			
	GPT-4o		**Llama3.1-8b**		**GPT-4o**		**Llama3.1-8b**		**GPT-4o**		**Llama3.1-8b**	
k	**B**	**BiTxt**	**B**	**BiTxt**	**B**	**BiTxt**	**B**	**BiTxt**	**B**	**BiTxt**	**B**	**BiTxt**
1	1.26	1.31	1.49	1.39	2.51	2.53	2.47	2.49	1.53	1.34	1.31	1.49
2	1.03	**1.27**	1.06	**1.51**	**2.73**	2.50	**2.43**	2.30	1.31	**1.51**	**1.57**	1.23
3	**1.14**	1.10	0.94	**1.23**	**2.71**	2.56	2.34	**2.66**	**1.47**	1.30	**1.46**	1.37
4	**1.13**	0.94	0.99	**1.06**	**2.61**	2.40	2.37	**2.80**	**1.43**	1.36	**1.43**	1.41

Table 3: Similarity Scores Averaged Across Examples. B represents the baseline score and BiTxt represents the score for BITEXTION algorithm. Higher similarity to the targets indicates better performance of the method. To indicate the cases where BITEXTION performs better compared to the baseline, we use **green highlight**. In constrast, **red highlight** indicates the cases where the baseline outperforms our method. We include the row $k = 1$ in the Table as a control, since no questions have been asked and no reponses have been received when we sample $d_1 \sim P_\theta(.|x_0 = (\mathbf{q}_0, \mathbf{r}_0, v) = (\emptyset, \emptyset, v))$ in the first iteration. For NL4OPT, we use a randomly selected subset of 70 examples from the 285 examples in the dataset to reduce computational cost of the experiments.

Limitations of Parameter Similarity: In parameter similarity, we observe that the baseline approaches outperform BITEXTION across all datasets and models. We hypothesize that this may be because, in sampled detailed descriptions, d_k, the parameters of the problem tend to be almost the same or very similar. Therefore, BITEXTION formulates questions that aim to discriminate between different objectives and constraints instead of focusing on the parameters of the problem. Consequently, BITEXTION fails to gather information about the parameters, whereas the baseline approaches occasionally ask questions about the problem parameters. We observe that these trends are present across all 3 datasets but are more pronounced in ComplexOR and NL4OPT compared to NLP4LP.

NLP4LP ($n = 65$)													
		Constraint Similarity				**Objective Similarity**				**Parameter Similarity**			
		GPT-4o		**Llama3.1-8b**		**GPT-4o**		**Llama3.1-8b**		**GPT-4o**		**Llama3.1-8b**	
Ratio	k	**B**	**BiTxt**	**B**	**BiTxt**	**B**	**BiTxt**	**B**	**BiTxt**	**B**	**BiTxt**	**B**	**BiTxt**
50-100%	1	1.22	1.67	1.56	1.22	4.00	3.78	4.22	4.11	1.56	1.56	1.67	2.00
	2	1.00	1.00	1.11	1.11	4.00	3.67	3.89	3.33	1.44	1.22	1.33	1.56
	3	0.89	0.89	1.00	2.00	4.00	4.00	4.22	3.89	1.56	1.78	1.78	1.22
	4	0.78	1.00	1.22	2.22	3.44	3.56	4.22	3.67	1.44	1.44	1.33	1.33
30-50%	1	1.21	1.29	1.21	1.57	3.29	3.29	3.50	3.50	1.07	1.07	1.36	1.29
	2	0.86	1.00	1.07	1.36	3.21	3.21	3.21	3.14	1.50	1.07	0.86	1.07
	3	1.00	0.57	1.36	0.79	3.14	3.36	3.43	3.29	1.07	1.00	1.07	0.71
	4	0.71	0.71	0.86	0.71	3.29	3.21	3.36	3.29	1.29	1.14	0.86	0.86
20-30%	1	1.16	1.21	1.26	1.26	3.11	2.95	3.16	3.37	1.21	1.26	0.74	1.21
	2	0.89	1.26	1.00	1.16	2.95	3.00	3.05	2.89	1.05	1.11	1.16	1.05
	3	1.00	1.16	1.32	0.74	2.74	3.16	3.05	2.84	1.05	0.84	1.00	1.05
	4	1.05	0.89	1.21	0.74	2.74	2.79	3.26	2.95	1.16	0.79	0.74	0.79
00-20%	1	0.79	0.74	0.84	0.68	3.26	3.05	3.26	3.16	1.21	0.95	1.05	0.95
	2	0.89	0.74	0.79	0.42	3.05	3.37	3.05	2.84	0.79	0.89	0.84	0.84
	3	0.84	0.63	0.68	0.32	2.95	3.16	3.05	2.74	1.16	0.84	0.79	0.47
	4	0.58	0.58	0.68	0.58	3.37	2.89	3.21	2.89	0.79	0.68	0.63	0.68
ComplexOR ($n = 20$)													
		Constraint Similarity				**Objective Similarity**				**Parameter Similarity**			
		GPT-4o		**Llama3.1-8b**		**GPT-4o**		**Llama3.1-8b**		**GPT-4o**		**Llama3.1-8b**	
Ratio	k	**B**	**BiTxt**	**B**	**BiTxt**	**B**	**BiTxt**	**B**	**BiTxt**	**B**	**BiTxt**	**B**	**BiTxt**
30-50%	1	1.00	1.00	2.00	1.00	3.00	2.00	3.00	3.00	1.00	2.00	2.00	1.00
	2	2.00	1.00	2.00	0.00	3.00	2.00	2.00	2.00	2.00	2.00	1.00	1.00
	3	3.00	2.00	2.00	0.00	1.00	2.00	2.00	1.00	2.00	2.00	1.00	2.00
	4	2.00	2.00	2.00	1.00	2.00	2.00	2.00	1.00	2.00	1.00	2.00	0.00
20-30%	1	1.33	1.33	1.33	1.33	3.67	3.33	3.67	4.00	2.33	1.67	2.33	2.00
	2	1.33	1.00	1.00	1.33	3.33	3.33	2.33	3.33	1.67	1.33	1.67	1.33
	3	1.00	1.33	0.67	1.00	2.67	3.33	4.00	3.00	1.33	1.67	0.67	1.00
	4	1.00	1.33	1.33	0.67	3.00	3.00	3.67	3.33	2.00	1.00	1.33	1.00
00-20%	1	1.38	1.25	1.19	1.31	3.69	3.81	3.38	3.81	1.44	1.19	1.25	1.19
	2	1.38	0.88	1.00	0.62	3.44	3.06	3.50	3.50	1.06	1.00	0.81	0.75
	3	1.06	0.62	1.62	0.75	3.94	3.56	3.69	3.50	0.69	1.00	1.19	0.62
	4	0.94	0.56	0.94	0.62	3.69	3.50	3.75	3.50	1.00	1.00	0.81	0.62
NL4OPT ($n = 70$)													
		Constraint Similarity				**Objective Similarity**				**Parameter Similarity**			
		GPT-4o		**Llama3.1-8b**		**GPT-4o**		**Llama3.1-8b**		**GPT-4o**		**Llama3.1-8b**	
Ratio	k	**B**	**BiTxt**	**B**	**BiTxt**	**B**	**BiTxt**	**B**	**BiTxt**	**B**	**BiTxt**	**B**	**BiTxt**
30-50%	1	1.57	2.14	2.00	2.00	2.00	2.29	2.29	2.14	1.14	1.43	2.29	1.71
	2	1.57	1.57	2.57	2.29	2.43	2.43	2.43	2.14	1.43	2.14	1.71	2.14
	3	1.86	1.43	1.86	2.00	2.43	2.29	2.29	2.29	1.71	2.43	2.00	2.00
	4	1.86	1.57	1.86	1.43	2.57	2.71	3.29	2.43	1.57	2.00	2.29	2.00
20-30%	1	1.39	1.68	1.54	1.46	2.93	2.79	2.96	3.21	1.89	1.54	1.64	1.50
	2	1.21	1.14	1.57	1.43	2.96	2.89	2.54	3.00	1.57	1.68	1.39	1.71
	3	1.29	1.14	1.39	1.32	3.11	2.79	3.18	3.00	1.71	1.61	1.39	1.43
	4	1.04	1.04	1.07	0.89	3.07	2.79	3.32	2.64	1.57	1.68	1.50	1.57
00-20%	1	1.09	1.20	1.14	1.06	2.29	2.26	2.14	2.06	1.31	1.11	1.20	1.14
	2	0.77	0.89	1.26	0.94	2.60	2.06	2.09	2.17	1.09	1.37	1.00	1.23
	3	0.89	0.69	0.97	0.74	2.46	2.00	2.31	2.26	1.23	1.14	1.23	1.06
	4	1.06	0.83	0.89	0.89	2.26	1.97	2.29	2.20	1.29	1.11	1.17	1.06

Table 4: Scores for subsets of the NLP4LP, ComplexOR, and NL4OPT datasets with different vague to detailed description length ratios.

C Supplementary Figures

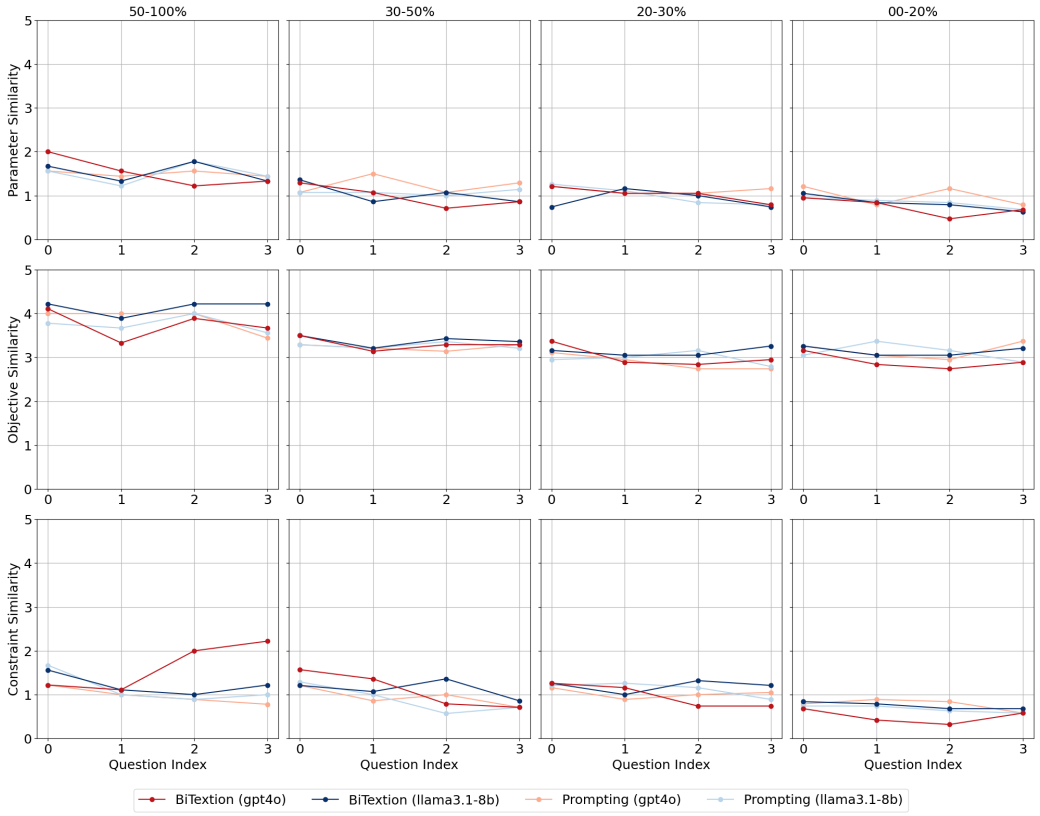

Figure 4: Scores for subsets of the NLP4LP Dataset with different vague to detailed description length ratios. $(n = 65)$.

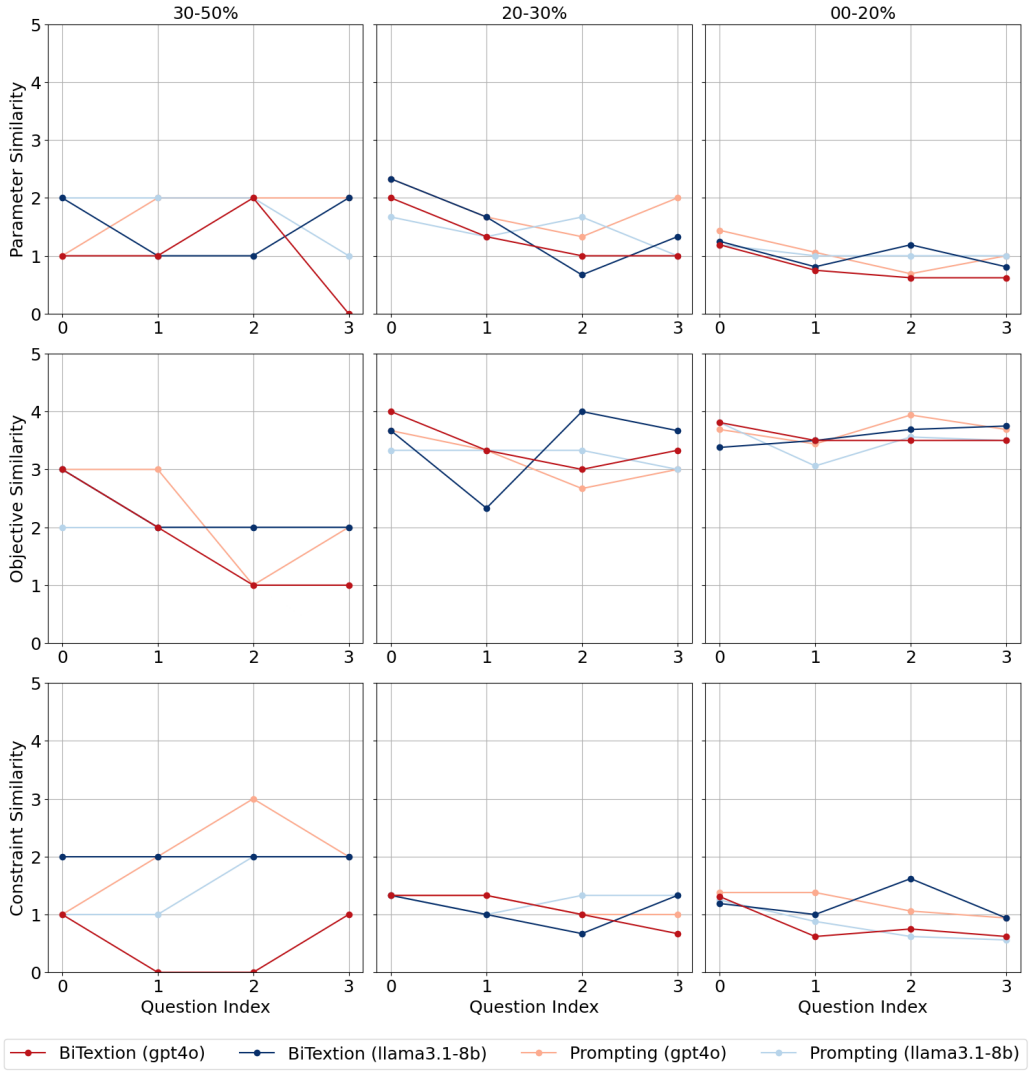

Figure 5: Scores for subsets of the ComplexOR Dataset with different vague to detailed description length ratios. $(n = 20)$.

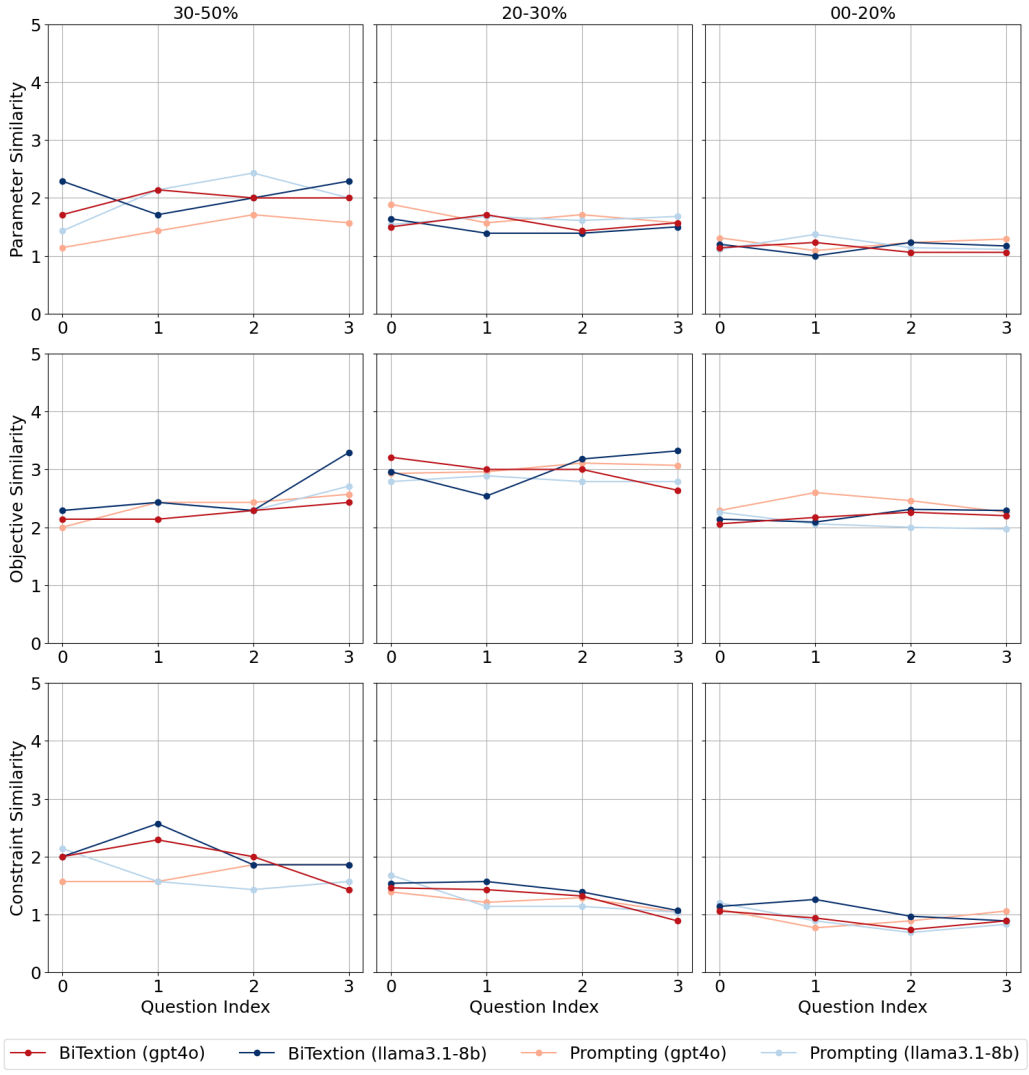

Figure 6: Scores for subsets of the NL4OPT Dataset with different vague to detailed description length ratios. $(n = 70)$.

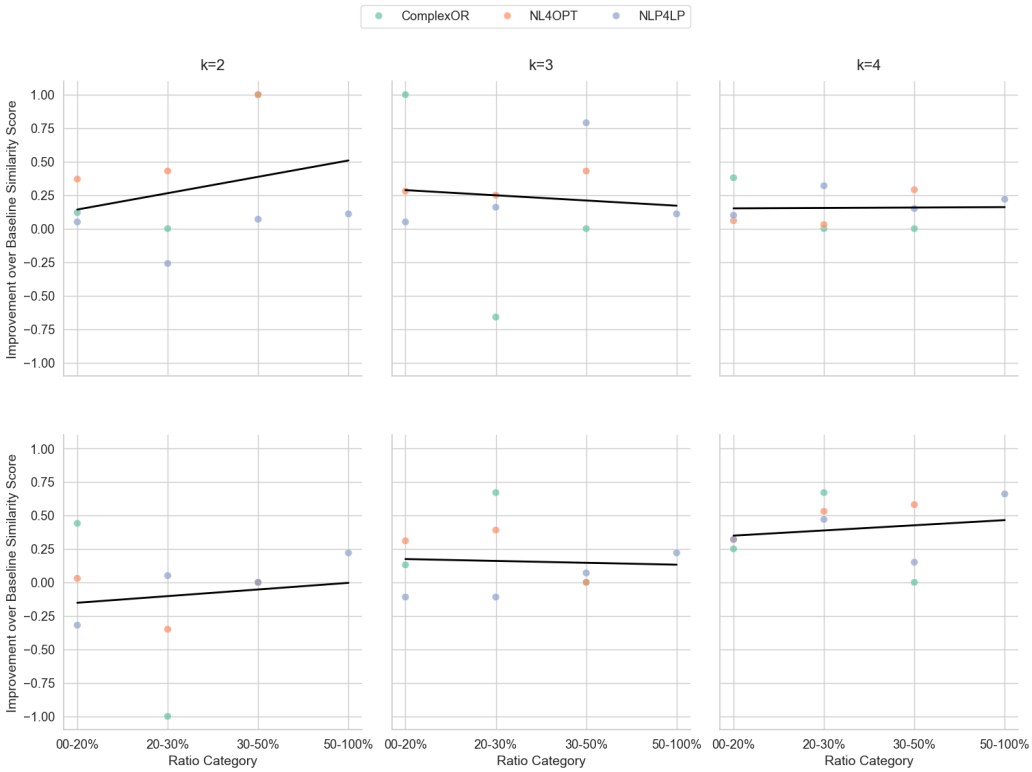

Figure 7: Regression Analysis. First row shows the improvement over the baseline constraint similarity. Second row shows the improvement over the baseline objective similarity. Plots are generated using the Llama models. Parameter similarity is excluded due to the lack of improvement over the baseline.