LEETPROMPT: Leveraging Collective Human Intelligence to Study Large Language Models

Sebastian Santy1 Ayana Bharadwaj1 Sahith Dambekodi2 Alex Albert1 Cathy Yuan2 Ranjay Krishna1

Abstract
Writing effective instructions (or prompts) is rapidly evolving into a dark art, spawning websites dedicated to collecting, sharing, and even selling instructions. Yet, the research efforts evaluating large language models (LLMs) either limit instructions to a predefined set or worse, make anecdotal claims without rigorously testing sufficient instructions. In reaction to this cottage industry of instruction design, we introduce LEETPROMPT: a platform where people can interactively explore the space of instructions to solve problems. LEETPROMPT automatically evaluates human-LLM interactions to provide insights about both LLMs as well as human-interaction behavior. With LEETPROMPT, we conduct a within-subjects user study (N = 20) across 10 problems from 5 domains: biology, physics, math, programming, and general knowledge. By analyzing 1178 instructions used to invoke GPT-4, we present the following findings: First, we find that participants are able to design instructions for all tasks, including those that problem setters deemed unlikely to be solved. Second, all automatic mechanisms fail to generate instructions to solve all tasks. Third, the lexical diversity of instructions is significantly correlated with whether people were able to solve the problem, highlighting the need for diverse instructions when evaluating LLMs. Fourth, many instruction strategies are unsuccessful, highlighting the misalignment between participant’s conceptual model of the LLM and its functionality. Fifth, participants with prompting and math experience spend significantly more time on LEETPROMPT. Sixth, we find that people use more diverse instruction strategies than these automatic baselines. Finally, LEETPROMPT facilitates a learning effect: participants self-reported improvement as they solved each subsequent problem.

1. Introduction
We are witnessing a Cambrian explosion of research in large language models (LLMs). LLMs have progressed from simply “understanding language” to assisting with problems in biology, solving math problems, answering general knowledge questions, and even writing code (Brown et al., 2020; Chowdhery et al., 2022; Rae et al., 2021). Given these newfound abilities, tracking how well these models work for different tasks is becoming increasingly difficult. In response, large benchmarking attempts, including GLUE (Wang et al., 2018), SuperGLUE (Wang et al., 2019), BigBench (Ghazal et al., 2013), and HELM (Liang et al., 2022), have been proposed. However, we argue that they remain limited since they evaluate models using a fixed set of LLM “instructions”, or what we often colloquially refer to as “prompts” (Liang et al., 2022). This limitation is highlighted by recent manuscripts that break away from such benchmarks and resort to showcasing LLM capabilities (e.g. specifically GPT-4’s) using anecdotal interactions between researchers exploring the space of instructions (Bubeck et al., 2023; Qin et al., 2023).

While the need to study how people invoke LLMs has always been at the center of today’s discourse, in the context of human-AI alignment (Christian, 2020; Gabriel, 2020), there is a dearth of non-anecdotal evaluations of LLMs with real human interactions. Such studies are particularly vital since the difficulty of finding effective instructions has led to websites and forums dedicated to collecting and sharing instructions (e.g. PromptHero, Arthub.ai, Reddit/ StableDiffusion). There are also online marketplaces for purchasing and selling useful instructions (e.g. PromptBase).

To enable a rigorous evaluation with real human interactions, we present LEETPROMPT. LEETPROMPT is an online platform populated with problems that users can attempt to solve by invoking LLMs with custom instructions; it inherits its name from LeetCode, where users can attempt
to solve similar problems by writing custom code (Figure 1). LEETPrompt is a dual objective platform similar to re-CAPTCHA (Von Ahn et al., 2008) and other collective human protocols (Kiela et al., 2021): On one hand, users use LEETPrompt to solve problems with LLMs; meanwhile, LEETPrompt automatically gathers evaluation metrics and user-behavior insights for researchers. On LEETPrompt, users also have the ability to add new problems, allowing the platform to organically grow. By granting people agency, our platform leverages collective human intelligence (Malone & Bernstein, 2022) to study which problems are unsolvable by language models and simultaneously provides user statistics on their interaction behavior.

To showcase the utility of LEETPrompt, we run a within-subjects evaluation where we invite participants to solve problems spanning 5 domains: Biology, Physics, Math, Programming, and General Knowledge. We chose these domains as representative of current LLM evaluations (Hendrycks et al., 2020; Patel et al., 2021; Sap et al., 2019; Tafjord et al., 2019; Sap et al., 2022; Moghaddam & Honey, 2023; Taylor et al., 2022; Chen et al., 2021; Liu et al., 2023). We invited a set of 4 initial users with prior experience evaluating and interacting with LLMs to write problems for LEETPrompt. They were also asked with problem setting: generating a set of problems and their corresponding private test cases, where the test cases serve the same purpose as those used in software development. In total, the problem setters populated LEETPrompt with 101 questions. From this set of questions, we sample 10 questions for our user study, such that there are 2 questions per domain. We recruit 20 participants, with varying education background, experience with LLMs/programming, and demographics. Each participant attempts to solve the 10 sampled problems using GPT-4.

By quantitatively analyzing and qualitatively coding the interactions across 1178 invocations of GPT-4, we make the following observations: First, we find that participants are able to design instructions for all tasks, including those that problem setters deemed unlikely to be solved. Sometimes, incorrect instructions can still result in correct LLM behavior. Second, all automatic mechanisms fail to generate instructions to solve all tasks, including zero-shot, few-shot, zero-shot CoT (Kojima et al., 2022), few-shot CoT (Wei et al., 2022), and auto-CoT (Zhang et al., 2023b). Third, the lexical diversity of instructions is significantly correlated with people’s ability to solve the problem, highlighting the need for diverse instruction strategies. Fourth, many instruction strategies are unsuccessful, highlighting the misalignment between the participant’s conceptual model of the LLM and its functionality. Fifth, participants with experience on instructing language models, and with math spend significantly more time on LEETPrompt. Sixth, we find that people use more diverse instruction strategies than these automatic baselines. Finally, LEETPrompt facilitates a learning effect: participants self-report solving subsequent problems faster and with a better strategy.

2. Related Work

We design LEETPrompt by drawing on ideas from existing benchmarks, recent anecdotal evaluations, and collective intelligence literature.

Benchmarks in machine learning. Reaching human performance on influential benchmarks is often viewed as a key milestone for a field (Ruder, 2021). For instance, AlphaFold’s superior performance on the CASP 14 competition marks a major scientific advance in the field of structural biology. Early benchmarks in machine learning include Switchboard (Godfrey et al., 1992) and MNIST (LeCun et al., 1998), which have been followed by ImageNet (Rusakovsky et al., 2015), SQuAD (Rajpurkar et al., 2016), and SNLI (Bowman et al., 2015). These benchmarks have rallied computer vision, natural language processing, and other domains of machine learning around a set of common goals. With the onset of LLMs, larger benchmarks such as GLUE (Wang et al., 2018; 2019), Eval Harness (Gao et al., 2021), BigBench (Ghazal et al., 2013), MMLU (Hendrycks et al., 2020), and HELM (Li et al., 2022) have helped define progress.

A trend towards anecdotal evaluation. Moving away from benchmarks, anecdotal proof of these models’ newfound powers is one of the most recent trends. “Sparks of AGI”, for example, shows numerous examples suggesting that GPT-4 pushes the needle towards artificial general intelligence (Bubeck et al., 2023). Many other papers have followed a similar pattern reporting on manually curated data points exhibiting surprising performance of language models (Qin et al., 2023) and laying big claims such as language models possessing a theory of mind (Kosinski, 2023). Without sufficient test cases for each claim and without an exhaustive exploration of possible input instructions, these claims require further investigation (Wei et al., 2022; Kojima et al., 2022). LEETPrompt serves as a platform where researchers can design problems, generate sufficient test cases, and allow a collective of users to explore the space of instructions.

Collective intelligence systems. Leveraging people’s collective intelligence has long since served the machine learning community. Collective intelligence, in the form of microtask crowdsourcing, catalyzed the community’s ability to annotate training (Russakovsky et al., 2015) and curate evaluation data (Zhou et al., 2019). Dual objective platforms, such as re-CAPTCHA (Von Ahn et al., 2008), simultaneously serve as a spam filter while also passively collecting annotations for vision and language tasks. Games-with-a-
LEETPROMPT: Leveraging Collective Human Intelligence to Study Large Language Models

Figure 1. LEETPROMPT is a platform where users can explore the space of instructions to solve tasks with LLMs.

A. Problem description: This panel contains a description of the problem that needs to be solved. It often contains examples of plausible inputs and outputs.

B. Interaction interface: Here, users will write instructions, change model hyper-parameters, and evaluate their instructions against public as well as private test cases.

C. Writing instructions: Here lies the text interface where users write their instructions. The token \[\text{INPUT}\] identifies where test cases will insert inputs. Most problems already contain a starter instructions to help users get started.

D. Model hyper-parameters: This panel shows which model is being used and the corresponding modifiable model hyper-parameters.

E. Test and submit: Users can test their instructions against a custom input by clicking Test or submit their instructions to be evaluated against the blind test cases by clicking on Submit.

F. Problem details and submissions: Users can check overall problem statistics, load past submissions, and compare their performance against other users.

Purpose is another class of systems where players derive casual enjoyment while playing games that contribute useful data for image labeling (Von Ahn & Dabbish, 2004), image segmentation (Von Ahn et al., 2006), commonsense reasoning (Von Ahn, 2006), math (Guess the correlation) and even protein folding (FoldIt). Lab-in-the-Wild projects similarly provide users with results from a psychology test, exchanging curiosity for their efforts (Reinecke & Gajos, 2015). Citizen science projects such as Stardust (Westphal et al., 2005) and Zooniverse (Simpson et al., 2014) host projects in a range of fields including astronomy, ecology, cell biology, humanities, and climate science. We draw inspiration from these methods to design LEETPROMPT’s dual use of LLMs evaluation while also studying user interaction behavior.

Human-driven evaluation. Recent work advocates for evaluating LLMs with human interactions (Lee et al., 2022). While they primarily leverage human interactions to analyze LLM outputs (Lee et al., 2022), we propose the opposite: we explore human interactions with LLM input instructions. Today, human-driven evaluations determine whether text is human- or machine-generated (Dugan et al., 2023) and measure how well generated images reflect human text inputs (Kirstain et al., 2023). Corporations have also started efforts into outsourcing their “red-teaming” evaluation to people in the wild (Ganguli et al., 2022; OpenAI, 2023). Our contribution can also be seen as a mirror opposite of Dynabench (Kiela et al., 2021), which incentivizes users to provide adversarial data to break model behavior; LEETPROMPT users produce instructions to most effectively solve problems.

3. LEETPROMPT

Our research introduces LEETPROMPT: an online platform where users are presented with problems from multiple domains and tasked with writing instructions that an LLM can use to solve the task. We hope to release LEETPROMPT publicly at a later date to test its utility at scale.

Designing the LEETPROMPT workflow. LEETPROMPT’s core functionality is inspired by online code judging systems such as LeetCode, TopCoder, and CodeForces. The platform contains a list of problems that users choose to tackle. Each problem also has a leaderboard indicating how many users have already attempted to tackle this problem, who achieved the best performance, and with how many submissions. Once the user chooses a problem, they are given the problem’s description as well as a few examples

1https://artwhisperer.io/
of inputs and outputs outlining expected behavior. With this information, users must write instructions that steer a large language model toward solving the given problem. Users can also change the model’s hyperparameters.

Once they’ve finished writing the instruction, users can test its utility against their own custom inputs. Custom inputs allows users to iteratively explore how their changes affect LLM behavior. Once satisfied, they can submit their final instruction to be evaluated against our hidden test cases. These private test cases are never revealed to the user; instead, they receive a percentage score indicating how many of the private test cases passed. To discourage rapid-fire submission and hill-climbing behavior on the private test cases, the platform provides a time-out after 3 submissions for 5 minutes. LEETPROMPT also records how many times the user submits instructions, and uses that information to rearrange the leaderboard to tactically encourage fewer submissions.

**Designing the LEETPROMPT’s interface.** LEETPROMPT is designed to be simple and promote an intuitive user experience (see Figure 1). We recruit UX designers to join our team and run studies using mockups of the interface to identify common pitfalls. We face the following design challenge: creating a platform that is intuitive for users who are not familiar with the process of writing instructions for LLMs. Drawing on design theory (Nielsen, 1995), the following principles are used: (1) Standardizing the interface to match with existing/popular LLM platforms and (2) employing recognition over recall so users know the full extent of the platform’s affordances through a quick skim. The appendix contains older versions of the interface.

**Evaluating user-generated instructions.** Figure 1(C) contains a screenshot of a possible instruction that a user might write to solve a problem. Users are expected to place an `[ [ INPUT ] ]` token somewhere in their produced instruction. LEETPROMPT evaluates instructions using the following:

$$\text{accuracy} = 100 \times \frac{1}{N} \sum_{i=1}^{N} 1[y_i == \text{LLM}((\text{Instruction}; [ [ \text{INPUT} ] ] = x_i))]$$

where \((x_i, y_i) \in [1, \ldots, N]\) are \(N\) private test case (input, output) pairs. (Instruction; `[ [ INPUT ] ] = x_i`) replaces the input token in the instruction with test case input \(x_i\). \(1[]\) is the indicator function. LEETPROMPT contains a set of common regexes to extract the desired output for the problem from the LLM’s generation.

**4. User study**

In the following section, we outline our user study design in which we study how people interact with LLMs to solve problems.

**Domain selection.** Language models were tested on several different problems. Pre-LLMs, most language systems focused on tasks such as sentiment analysis (Mäntylä et al., 2018), part-of-speech tagging (Chiche & Yitagesu, 2022), machine translation (Stuhlberg, 2020), entailment (Mitkov, 2022), and speech processing (Nassif et al., 2019). Since LLMs, models are now evaluated on simple mathematical and commonsense reasoning (Hendrycks et al., 2020; Patel et al., 2021; Sap et al., 2019; Tafjord et al., 2019), causal reasoning (Kiciman et al., 2023) and theory of mind (Sap et al., 2022; Moghaddam & Honey, 2023), basic sciences (Taylor et al., 2022), programming (Chen et al., 2021; Liu et al., 2023), and even law exams (Zhong et al., 2023; OpenAI, 2023). We choose a subset of these domains as the test bed for our user study: Biology, Physics, Math, Programming, and General Knowledge.

**Problem setters.** LEETPROMPT grants people agency by empowering them to use our platform to study new problems they might struggle to solve with LLMs. We invite a small set of 4 initial users with prior experience interacting with and using LLMs to LEETPROMPT. We ask these users to design problems: they collectively write a set of 101 problems, each with a set of 5 private test cases. Once the problems are designed, the problem designers are asked to try and solve each problem and to assign a subjective difficulty based on their experience. Each problem is categorized into one of three difficulty levels using majority vote. Easy problems are those that can be solved by utilizing a simple formula or equation. Medium problems contain difficulty either in the complexity of the formulas or parsing the inputs and describing the task. They sometimes require domain knowledge or some background information that can simplify the problem. Hard problems involve complex logic or edge cases.

From this set, we sample 10 problems such that there are 2 problems for each of the 5 aforementioned domains. Each participant in our user study sees these 10 problems. Table 1 describes these 10 problems. In total there are 3 easy problems, 5 medium problems and 2 hard problems. The 2 hard problems were chosen because the problem setters were not able to find an instruction that worked for all the test cases. Problems are randomly ordered for each user.

**Example problem.** Figure 1 lays out one of our Biology problems and a user-generated instruction. It presents the participant with a food chain in the form of \(X \rightarrow Y\) implies that species \(Y\) is at a higher tropic level and, therefore, eats
species $X$. The user is also presented with two species sampled from that food chain. The problem asks the user to determine whether an increase in the population of the first species will lead to an increase or decrease in the second. Example 1 in the problem description presents the following food chain: kelp $\rightarrow$ sea urchin $\rightarrow$ otter $\rightarrow$ orca. The two species presented are “sea urchin” and “kelp”. Since “sea urchins” eats kelp, an increase in the “sea urchin” population leads to a decrease in the “kelp” population. This is a hard problem.

The user-generated instruction uses pseudocode to inform the model that if the first species is at a higher trophic level than the second, the population of the second species will decrease, and vice versa. Even though this logic works for “sea urchin” and “kelp”, this pseudocode is incorrect when the two species are “otter” and “kelp”. Since there is a skip connection between “otter” and “kelp”, an increase in “otters” will result in a decrease in “sea urchins”, which will in turn result in an increase in “kelp”.

**Study protocol.** We design a within-subjects study to explore the utility of LEETPROMPT. To reiterate, the study measures LEETPROMPT’s utility along two simultaneous goals: Investigating 1) whether the platform supports participants in writing effective instructions to solve problems with LLMs and 2) providing insights into how people interact with LLMs as they explore the space of instructions.

The study begins by asking participants to sign a consent form that outlines that their interactions will be stored and used for this research contribution. Next, they are asked an initial survey to record their demographics, their background education, and their experience working with LLMs or programming more broadly. Next, they are shown instructions for using the interface and provided resources that they can reference for advice on good instruction design. After this, the participant is shown a demo of the interface, which highlights and explains the various panes in the interface. To help familiarize themselves with the interface, users are provided a sandbox interface with a toy starter problem that they can test the functionality afforded by the platform. Next, the participants are presented with the 10 sampled problems and asked to attempt solving each one. We set expectations that the study will likely take about 1 hour. The study ends with a final survey where we collect open-ended responses describing their experience.

**Participant selection.** All of our participants are recruited locally from the authors’ city/town. We only include participants that speak English, spent at least 1 hour working on problems, and submitted at least one instruction to each of the 10 problems. All participants receive a payment of $15 for their time with no bonuses. We provide no other incentives other than intrinsic motivation for subsequent attempts at designing better instructions for each problem. Regardless, we find that each user tried an average of 5.8 instructions per question.
Instructions: Divide the volume by 100. Give me the answer in the form \[ \text{output} \times 100 \].

Let's think step by step. We are given the initial pressure (P1) as 3 atm, the initial temperature (T1) as 300 K, and the final...

Figure 2. Visualizing the space of Auto-CoT and LEETPROMPT Instructions: 2D Principal Component Analysis (PCA) of embeddings of Auto-CoT and collected instructions from LEETPROMPT. Auto-CoT instructions are marked as purple X. LEETPROMPT instructions are marked as dots with colors representing its solvability: "Test" instructions are colored gray. For "Submitted" instructions, red color indicates that they failed all testcases, yellow indicates they passed 1-4 testcases, and green indicates that they passed 5 testcases. Instructions are specifically shown for two problems each from each domains to illustrate the different strategies used by participants and whether they were successful.

Measured variables. To evaluate instructions, we use two primary objective performance measures: solvability and diversity. Solvability measures the percentage of test cases that pass when using a specific instruction. Diversity measures how creative people are in writing instructions. Diversity provides insights into the different approaches that a participant takes to solve each problem. It allows us to evaluate the claim that diverse instructions and ensembling leads to significant boosts in LLM performance (Yoran et al., 2023). Since diversity of instructions can be measured in multiple ways, we measure the following: (1) Lexical Diversity (LD), Semantic Diversity (SD), and Approach Diversity (AD).

LD uses Repetition Rate (RR) (Cettolo et al., 2014; Bertoldi et al., 2013) to measure diversity in lexicons between a participant’s instruction and the original wording used in the problem description. Specifically, LD = 1 – RR. LD is 0 if all lexicons appear more than once and 1 if all lexicon are new.

SD measures the diversity in strategies used by a user when tackling a specific problem. We use text+code embeddings (text-embedding-ada-002) to encode each instruction and visualize their first two principle components for qualitative visual analysis. For a quantitative analysis, SD is the variance for the first principal component.

AD measures the percentage of instruction strategies used, where strategies are qualitatively coded by 3 independent coders (see Table 2 for codes and appendix for full details).
The qualitative codes reflect common instruction strategies in literature: prime the model to behave like a domain expert (Argyle et al., 2022), use example input/outputs (Wang et al., 2022a), use CoT prompting (Wei et al., 2022; Kojima et al., 2022; Wang et al., 2022b; Ye & Durrett, 2022; Turpin et al., 2023), use self-help (Press et al., 2022; Zelikman et al., 2022) and the lexical structure (Fu et al., 2022), and use pseudo-code (Zhang et al., 2023a).

**Automatic instruction mechanisms.** We compare user-generated instructions with mechanisms used in existing benchmarks. Zero-shot (0s) uses the problem description and no other information. Zero-shot CoT (CoT) includes a phrase which requests the LLM to reason the steps aloud (Kojima et al., 2022). Few-shot variants (N-shot Ns where N = 1, 2, 3, 4) append N-example input/output pairs. We also test with advanced auto-prompting methods, such as Auto-CoT (Zhang et al., 2023b) which invokes multiple CoT reasoning steps along with 10 input/output examples, and Synthetic prompting (Shao et al., 2023) which tunes existing prompts.

**Models used.** LeetPrompt supports any LLM that affords an API access to invoke the model. We allow participants to use any of the following models: GPT-4, GPT-3.5, and GPT-3. However, all participants chose to use GPT-4 for all their attempts. In the appendix, we retroactively evaluate their instructions on the other models as a point of comparison.

### 5. Results

By analyzing the 1178 user-generated instructions and their feedback from the survey we find the following: To start, participants are able to generate instructions to solve all the problems, including the hard questions that the problem setters had deemed unlikely to be solved. Sometimes, incorrect instructions can still result in correct LLM behavior. Second, automatic mechanisms are not able to solve all problems. Third, the use of diverse instructions is strongly correlated with solvability, indicating a need for diversity in instructions in existing benchmarks. Fourth, there is a misalignment between user expectations of LLMs and the reality of how interpreted user instructions; participants stated that they were unprepared for the challenges they faced. Fifth, participants with more experience in prompting or math spent more time on LeetPrompt. Sixth, we find that people uncovered a diverse array of instruction strategies than these automatic baselines. Finally, participants self-reported a learning effect and better strategy use as they progressed through the study.

**Participants found solutions even though the problem setters were not able to solve them.** As shown in Table 1, all problems were solved by at least one participant. The problem setters’ difficulty categorization is strongly correlated with how many test cases participants passed (r(19) = .74, p = .01) and what percentage of participants were able to solve all 5 private test cases (r(19) = .84, p = .001). For “Digit Sum”, only one participant was able to solve the question. Surprisingly, the successful solution involved a specific re-wording of the question that improved the problem’s clarity and thus made it easier for the model to understand and solve the problem. Only through collection action is LeetPrompt able to identify this solution. Similarly, half the participants were able to solve “Food Chain”. Surprisingly, one winning instruction was a logically incorrect reasoning step that somehow still passes the test cases (shown in Figure 1). This adds more support to concurrent work, announced this week, which also finds that unfaithful reasoning steps improve LLM performance (Turpin et al., 2023).

**None of the automatic mechanisms were able to find instructions to solve all the problems** (Table 4). Furthermore, the diversity metrics (LD and SD) are both significantly smaller. A low LD implies that these mechanisms do not deviate from the original problem description. We visualize the lack of diversity of Auto-CoT in Figure 2, which visualizes the first two principal components of the SD instruction embeddings for each problem.

**Lexical diversity of instructions is significantly correlated with the number of passed test cases** (p(19) = 0.71, p < 0.01). (Table 5). This mirrors prior work (Li et al., 2022) which found that having diverse reasoning allowed for better LLM performance. This result suggests that large LLM benchmarks should rethink the use of a small fixed set of instructions. From Table 4 the best automatic mechanism, Auto-CoT, has lower LD than LeetPrompt participants (0.26 vs 0.49) and also passes fewer number of test cases (45 vs 50) which further proves this point.

**Participants struggled when debugging unexpected model behavior.** Participants reported a sense of confusion between their expectations versus how the model worked. Figure 2 shows visually that there exists entire clusters of instructions that do not solve the problem. For example, math-related strategies to solve the two math problems didn’t work while program-related strategies did. Participants complained that the model “would do math incorrectly” (P42). Similarly, using domain-specific information, such as using Ohm’s Law to solve the “Resistance is Futile” physics question failed while using math-related instructions sometimes succeeded. Even so, one participant exclaimed, “what was also strange was that the numerical answer the LLM provided could change based on seemingly trivial additions to the prompt: “I would perturb prompts in small ways that might unexpectedly change the output’” (P37).

**Participants with experience with LLMs, or an education in math spent significantly more time writing**
instructions \(p(19) = 0.59, p < 0.05\) and \(p(19) = 0.54, p < 0.05\). From Table 5, those with either experience with LLMs or prompting were tending towards significance for solving more problems \(p(19) = 0.42, p < 0.1, p(19) = 0.43, p < 0.1\). From Table 5, there is no strong correlation between the participant’s domain knowledge with solving problems in that domain. Taken together, these two findings suggest that knowing how to instruct the model can be more important than domain knowledge about the problem.

Participants uncovered a variety of instruction strategies to solve problems. Each column in Figure 2 is a different domain. Interestingly, the clusters appear to follow a similar pattern between the two rows, implying that people use similar strategies for problems within a given domain.

On average, natural language worked better than pseudocode. By qualitatively analyzing the clusters in Figure 2), we find that participants who used pseudocode instructions generally wrote longer instructions to account for all possible edge cases, similar to writing exceptions in software. Debugging these instructions was easier even though writing the complex instruction took longer. For those who wrote natural language instructions, their instructions were shorter, and more successful. The one exception was the Physics domain, for which simplifying the domain-specific knowledge about physics into arithmetic equations worked the best. Unfortunately, participants with a background in programming mentioned that they found it difficult to break away from writing programming instructions, “Was thinking in a programming language/code and found it hard to translate that to an instruction” (P57) and, “Wasn’t able to express programmatic reasoning” (P42).

Explicit, detailed instructions do better but instruction structure can also affect performance. Participants learned a need for clarity and to “give more direct and structured commands” (P34) and be “extremely explicit and factual knowledge and did not require explicit reasoning. In

---

### Table 5. Pearson’s correlation coefficient between participant attributes (demographic, background, and experience) and the maximum number of test cases passed and the time taken for each problem. ‘.’ indicates trending towards significance \((p < 0.1)\) and ‘*’ denotes significance \((p < 0.05)\). **Pass** is the average number of testcases passed and **Time** is the avg. time taken between first and last interaction with the problem.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Biology</th>
<th>Physics</th>
<th>Math</th>
<th>Programming</th>
<th>General</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age</td>
<td>0.23</td>
<td>-0.23</td>
<td>0.13</td>
<td>0.20</td>
<td>0.36</td>
<td>0.44</td>
</tr>
<tr>
<td>Experience</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Biology</td>
<td>-0.03</td>
<td>-0.11</td>
<td>-0.32</td>
<td>0.03</td>
<td>-0.41</td>
<td>-0.50</td>
</tr>
<tr>
<td>Physics</td>
<td>0.12</td>
<td>0.01</td>
<td>0.18</td>
<td>0.46</td>
<td>0.04</td>
<td>0.03</td>
</tr>
<tr>
<td>Math</td>
<td>0.18</td>
<td>0.25</td>
<td>0.01</td>
<td>0.31</td>
<td>0.10</td>
<td>0.30</td>
</tr>
<tr>
<td>LLM</td>
<td>0.09</td>
<td>0.51</td>
<td>-0.11</td>
<td>0.22</td>
<td>0.43</td>
<td>0.19</td>
</tr>
<tr>
<td>Prompting</td>
<td>0.43</td>
<td>0.13</td>
<td>0.21</td>
<td>0.45</td>
<td>0.35</td>
<td>0.35</td>
</tr>
<tr>
<td>Programming</td>
<td>0.27</td>
<td>0.18</td>
<td>-0.05</td>
<td>-0.58</td>
<td>0.35</td>
<td>0.25</td>
</tr>
</tbody>
</table>

---

### Table 4. Comparison of existing prompting approaches to prompts collected from LEETPrompt. **0s**: Zero-shot; **0s CoT**: Zero-shot Chain-of-Thought prompting; **1s, 2s, 3s, 4s**: 1,2,3,4 shot (or examples) prompting; **Auto-CoT**: Automatic Prompting Method; **Ours**: Prompts from LEETPrompt. P denotes the maximum number of testcases that the the given method was able to pass. As Auto-CoT and Ours have multiple prompts per problem, we report **LD & CD** which are the lexical and content diversity of prompts for each problem. We do not report **AD** for Auto-CoT since it defaults to using CoT as the main strategy for solving the problem.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Question</th>
<th>0s CoT P</th>
<th>0s CoT P</th>
<th>1s P</th>
<th>2s P</th>
<th>3s P</th>
<th>4s P</th>
<th>Auto-CoT P</th>
<th>Ours LD</th>
<th>Ours SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Biology</td>
<td>Water Potential</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.17</td>
<td>5</td>
<td>0.56</td>
</tr>
<tr>
<td>Biology</td>
<td>Food Chain</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.25</td>
<td>5</td>
<td>0.48</td>
</tr>
<tr>
<td>Physics</td>
<td>Ideal Gas Law</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>0.42</td>
<td>5</td>
<td>0.38</td>
</tr>
<tr>
<td>Physics</td>
<td>Resistance is Futile</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>0.34</td>
<td>5</td>
<td>0.68</td>
</tr>
<tr>
<td>Math</td>
<td>Consecutive Integers</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>0.34</td>
<td>5</td>
<td>0.68</td>
</tr>
<tr>
<td>Math</td>
<td>Digit Sum</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>0.31</td>
<td>5</td>
<td>0.46</td>
</tr>
<tr>
<td>Programming</td>
<td>Intersperse</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>0.40</td>
<td>5</td>
<td>0.51</td>
</tr>
<tr>
<td>Programming</td>
<td>Sort Numbers</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>0.37</td>
<td>5</td>
<td>0.49</td>
</tr>
<tr>
<td>Knowledge</td>
<td>Beatles Title</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>0.11</td>
<td>5</td>
<td>0.47</td>
</tr>
<tr>
<td>Knowledge</td>
<td>Theory of Mind</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>0.11</td>
<td>5</td>
<td>0.49</td>
</tr>
</tbody>
</table>

---

### Table 5.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Biology</th>
<th>Physics</th>
<th>Math</th>
<th>Programming</th>
<th>General</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pass</td>
<td>0.23</td>
<td>-0.23</td>
<td>0.13</td>
<td>0.20</td>
<td>0.36</td>
<td>0.44</td>
</tr>
<tr>
<td>Time</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

LEETPrompt: Leveraging Collective Human Intelligence to Study Large Language Models
Participants demonstrated and self-reported a learning effect as they progressed through the study. Multiple participants mentioned that they began adopting specific strategies as they completed more questions. “Yes, as I went through the problems I learned how to ask the model questions so it could help me answer the question. I also learned how to provide reasoning for the examples.” (P39) “I realized simplifying the problems greatly increased the reliability of the model, so I tried to rephrase questions to be as simple as possible” (P42).

6. Discussion, Future Work and Limitations

Diverse Instructions With the capability to collect diverse instructions from people by leveraging their creativity and desire to solve problems, there is potential for using them to train or “instruction-tune” language models (Ouyang et al., 2022). While initial efforts was towards collecting such instructions in controlled annotation setups from crowd-workers (Ouyang et al., 2022; Mishra et al., 2022), due to difficulty in collecting human-generated instructions, most of recent works resort to distilling large language models for generating instructions (Wang et al., 2022b) often to fine-tune smaller models (Peng et al., 2023; Taori et al., 2023). This presents a significant trade-off between quality (specifically, diversity which is crucial for language models (Yoran et al., 2023)) and quantity that distillation offers.

From our research, it is clear that users are clearly motivated to solve problems and have fun which helps in diminishing this trade-off between quality and quantity to a great extent. In addition to human-generated instructions being high quality, they highlight the human desiderata towards interacting with language models which can be used to align models closer to their preferences. LEETPrompts setup can be further utilized in places where it is becoming commonplace to use LMs for fixing LMs such as red-teaming (Perez et al., 2022), for example.

Micro-analysis of LLMs. LLM evaluation has largely relied on benchmarking, which was especially useful in the era of interpretable machine learning, where testing on large-scale datasets provided insights into how the algorithms would perform when deployed in the real world. The field is currently facing an inverse problem, in which the real-world utility of language models is obvious to many people, but the algorithm is a black-box, preventing understanding of its behavior. Furthermore, with the advent of web-scraped pretraining data, benchmark contamination has risen to become a common problem in the community. Through training data leakage dataset leakage degrade its utility, especially since simply looking at the numbers over large datasets is not very clear. Researchers have advocated for a change in evaluation, promoting protocols for micro-analysis—evaluating models on individual inputs or tasks (Olah et al., 2020). Behaviors and has been advocated since the development of complex neural architectures that function without an adequate understanding of what is going on under the hood (Olah et al., 2020). This is exacerbated by prompting as an input modality, which introduces confounders that are difficult to measure using standard evaluation approaches especially if done at a macro level.

LEETPrompts could be adapted to serve as a micro-analysis platform, allowing researchers to zoom into individual problems. The door to a different approach to evaluating these LLMs by assisting in zooming into specific tasks, strategies, and wordings used as a way to collectively understand the complex behaviors of these language models with the ability to share insights in a scientific manner.

Future directions and limitations In the future, LEETPrompts could serve purposes beyond evaluation. It could support users produce tutorials to teach others how to use LLMs; it can serve as a platform for the public to audit released models; it could also become integrated with existing benchmarks, like HELM, to provide micro-analysis evaluation of individual inputs; it can also track interaction patterns between people and LLMs over time. However, it does have its set of limitations: First, the user interactions from our study are unlikely to be indicative of LEETPromt’s usage in the wild with 20 participants with monetary compensation. Second, the demographics of participants mostly included people who have STEM knowledge and had at least heard of language models. Third, our problems did not explore open-ended tasks, creativity tasks, or complex tasks that require tool-usage. Regardless, LEETPromt’s results are encouraging: our results uncovered multiple user-behaviors by collectively organizing users around individual problems.

References


LEETPROMPT: Leveraging Collective Human Intelligence to Study Large Language Models


LEETPROMPT: Leveraging Collective Human Intelligence to Study Large Language Models


