arXiv:2408.05534v2 [cs.SE] 4 Feb 2025

Can LLMs Replace Manual Annotation of Software Engineering Artifacts?

Toufique Ahmed*†§, Premkumar Devanbu*, Christoph Treude‡ Michael Pradel§
*University of California, Davis, USA
†IBM Research, Yorktown Heights, New York, USA
‡Singapore Management University, Singapore
§University of Stuttgart, Germany

***Abstract*—Experimental evaluations of software engineering innovations, e.g., tools and processes, often include *human-subject studies* as a component of a multi-pronged strategy to obtain greater generalizability of the findings. However, human-subject studies in our field are challenging, due to the cost and difficulty of finding and employing suitable subjects, ideally, professional programmers with varying degrees of experience. Meanwhile, large language models (LLMs) have recently started to demonstrate human-level performance in several areas. This paper explores the possibility of substituting costly human subjects with much cheaper LLM queries in evaluations of code and code-related artifacts. We study this idea by applying six state-of-the-art LLMs to ten annotation tasks from five datasets created by prior work, such as judging the accuracy of a natural language summary of a method or deciding whether a code change fixes a static analysis warning. Our results show that replacing some human annotation effort with LLMs can produce inter-rater agreements equal or close to human-rater agreement. To help decide when and how to use LLMs in human-subject studies, we propose model-model agreement as a predictor of whether a given task is suitable for LLMs at all, and model confidence as a means to select specific samples where LLMs can safely replace human annotators. Overall, our work is the first step toward mixed human-LLM evaluations in software engineering.**

***Index Terms*—LLMs, human subjects, evaluation**

I. Introduction

Given that developer effectiveness heavily depends on good tools and processes, researchers constantly seek more and better automation in these areas. To mention just a few examples, there are now many research efforts [1], [2] aimed, *e.g.,* at automated code summarization, (i.e., techniques that generate a natural language summary for a given piece of code), at detecting bugs and other issues in a program [3], and at determining whether a warning produced by a static analysis tool is actually worth addressing [4].

But the value of such innovations ultimately is dependent on human judgment and practice. For example, a developer might read a generated English summary of a method, and then decide whether and how to use that method. As another example, a developer might look at a warning produced by a static analysis tool, and then decide whether to fix it or not. Because measuring human usefulness is difficult, many research efforts rely on proxy metrics. For the example of code summarization, a summary might rate a reasonable BLEU score, indicating at least some level of similarity to the original, human-written summary. However, even given a summary with a high BLEU score, can we be sure that it is clear, precise, relevant to the input code, and human-comprehensible?

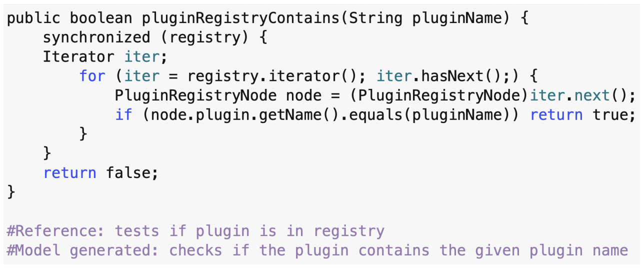
To answer such questions, researchers often use human subject-based evaluations, *e.g.,* when evaluating code summarization techniques [5], [6]. Besides code summarization, such evaluations are becoming increasingly common in many different settings; indeed in most cases where a tool is used to automate some aspect of software development, the output from the tool will be used by a human. Since the content and form of the output may affect how well a developer can use it, a human-subject evaluation is often vital.

But human-subject studies in software engineering are *costly*. For external validity, such studies demand both a representative sample of developers and a representative sample of relevant artifact samples. Each sample may need to be evaluated by multiple humans, to get more stable results. For example, Haque et al. [5], hired professional developers, at $60 per hour, to rate a total of 420 human and machine-generated code summaries, with three ratings for each sample to gauge inter-rater agreements. Due to the high costs, such evaluations are sometimes done using free or low-cost participants, such as students, which carries risks of the results not generalizing to professional developers. Yet, even performing a human-subject study with students can be quite time-consuming. Thus, the main motivation for researching new tools and techniques in software engineering, *viz.* the high cost of developers, is also a hurdle to the proper evaluation of such tools and techniques.

Given the ability of advanced large language models (LLMs), to rival human performance in a range of tasks [7]–[9], the question naturally arises: *Can we use LLMs to reduce the cost of human-subject studies in software engineering?* If LLMs can help, even if only partially, this could impact the practice of evaluation studies in software engineering. Thus, we, ask: When, and how, can human subject responses be safely replaced by LLMs, in a mixed human-LLM evaluation scenario?

This question has arisen in other disciplines, *e.g.,* psychology, linguistics, and medicine [10]–[13]. There are several reasons why the software engineering domain is particularly

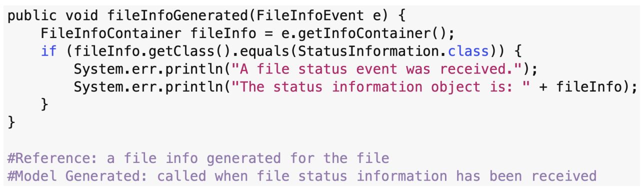
§Majority of the work was done when the author was a postdoctoral scholar at UC Davis.



```
public boolean pluginRegistryContains(String pluginName) {
    synchronized (registry) {
    Iterator iter;
        for (iter = registry.iterator(); iter.hasNext();) {
            PluginRegistryNode node = (PluginRegistryNode)iter.next();
            if (node.plugin.getName().equals(pluginName)) return true;
        }
    }
    return false;
}

#Reference: tests if plugin is in registry
#Model generated: checks if the plugin contains the given plugin name
```

(a) Sample where human raters agreed



```
public void fileInfoGenerated(FileInfoEvent e) {
    FileInfoContainer fileInfo = e.getInfoContainer();
    if (fileInfo.getClass().equals(StatusInformation.class)) {
        System.err.println("A file status event was received.");
        System.err.println("The status information object is: " + fileInfo);
    }
}

#Reference: a file info generated for the file
#Model Generated: called when file status information has been received
```

(b) A sample where humans disagreed

Fig. 1: Annotation tasks with different difficulty level.

interesting for this exploration: First, relative cost: LLMs can run hundreds of queries for the cost of using a single human subject for an hour. Second, LLMs now perform well on a wide range of software engineering tasks, and are routinely used in industry [7], [14]–[16]. Third, the impressive capabilities of LLMs at in-context learning [17], [18], suggest that with a few illustrative examples, LLMs can execute fairly complex software engineering tasks. Finally, software artifacts are arguably more complex than natural language artifacts: they can require deep knowledge of both application domain and programming; also, they comprise multiple formal & informal elements (*e.g,* code, summary, warnings).

This paper studies the possibility of substituting human subjects with LLMs when annotating software engineering artifacts. We study ten tasks from five datasets created by prior work, covering research ranging from requirements engineering to reasoning about static analysis warnings. We apply six state-of-the-art LLMs, including both open- and closed-source models, to these tasks, and then compare the LLMs' responses to those from human subjects.

We focus on tasks whose subjective, nuanced and context-dependent nature have traditionally required human judgment. This subjectivity is evident in the fact prior studies reveal considerable inter-rater *dis*agreement. Figure 1 shows two examples from code summarization, where human subjects are asked to rate the accuracy of generated summaries. In Figure 1a, all the three raters and the GPT-4 model rated "Strongly agree". However, other examples are inherently hard for humans, such as Figure 1b, where the three raters give different ratings: "Agree", "Strongly Disagree", and "Disagree", and GPT-4 agrees with the first rater. Our study asks: *Can powerful LLMs (partially) substitute for humans on inherently subjective annotation tasks?*. Our findings are:

- On some tasks, LLMs agree with human subjects about as much as human subjects agree with each other. For example, when judging whether an identifier name and the value it refers to are consistent [19], we observe a mean human-human inter-rater agreement of 0.52, and a mean human-model agreement of 0.49. This results suggests that LLMs can sometimes be a meaningful replacement for humans on *some* annotation tasks. But when, exactly?
- To help researchers decide whether an LLM may be suitable for a specific annotation task, we investigate the correlation between model-model agreement, which can be measured programmatically, and the ability of LLMs to meaningfully replace humans. We find model-model agreement strongly correlates with human-model agreement, suggesting that model-model agreement can be used to decide whether to involve LLMs in an evaluation.
- Some examples are easier to annotate than others. For a given example, can an LLM substitute for a human? We find that the LLM's confidence (its output probability) helps select examples that do not necessarily require human annotators. For example, for the task of judging whether a code snippet and its summary are similar, delegating 50% of the effort performed by one annotator to an LLM (selected based on LLM confidence), does not lead to a statistically significant change of the overall inter-rater agreement.

In summary, we make the following contributions.

- We offer a first study as to whether LLMs can replace human annotation on software engineering artifacts.
- We present a methodology designed to answer this question and empirical results from applying six LLMs to ten tasks addressed by humans in prior work.
- We propose a method for selecting which *tasks* are suitable for an LLM-augmented evaluation and which *samples* can be safely delegated to an LLM.

II. Background

In software engineering research, human-subject annotations provide insight into the potential impact of innovations on productivity or quality [20]; but the nuanced, subjective nature of these evaluations often requires multiple human raters to ensure consistency and reliability in the data [21].

Inter-rater reliability provides a vital gauge for the objectivity of the annotations [22]. It is commonly measured using statistical metrics such as Cohen's κ [23] or Krippendorff's α [24], which measure agreement among annotators by considering the possibility of pure-chance. Higher values suggest a strong, reliable, and robust agreement among annotators.

Annotations based on a single annotator carry a risk of personal biases and subjective interpretations, and lead to inaccurate, non-generalizable results [25]. In contrast, multiple annotators working independently can reveal an inherently shared understanding, which indicates that the annotations accurately capture the underlying phenomena being studied. However, achieving high inter-rater reliability is effortful, time-consuming, and challenging; human developer time is costly, and they often disagree!

LLMs, since they are trained on very large corpora, could help these challenges by potentially providing more consistent annotations which normatively reflect "most" developers. They could reduce the biases introduced by human annotators, LLM-based automation of annotations could make it feasible to analyze datasets that were previously too large or complex to annotate manually. In addition, software engineering data is dynamic and constantly evolving, requiring tools that can quickly adapt to new information. LLMs, with their ability to learn from and adapt to new data, could provide up-to-date annotations, complementing the work of human annotators.

To effectively incorporate LLMs into qualitative annotation, it is essential to establish a structured process. This includes deciding when and how to use LLMs, integrating them into existing workflows, distributing tasks between humans and LLMs, and determining the degree of manual work required. This paper addresses these considerations through four research questions (See subsection III-C), focusing on annotations with pre-existing categories derived from related work or developed in an initial research phase. Future work will explore the potential of LLMs for annotating qualitative data without pre-existing categories (i.e., open coding) within a software engineering context.

III. Methodology

We now describe the tasks, datasets, the models under consideration, our research questions, and the methodology to answer these questions.

A. Tasks & Datasets

We select five datasets from previous work, which together include 10 human-annotation tasks. For some datasets, multiple tasks were assigned to the raters. For example, in the code summarization dataset, raters were given four tasks: rating accuracy, adequacy, similarity, and conciseness. Similarly, for the semantic similarity dataset, raters were assigned three different rating tasks. This is why we have five datasets or major tasks but a total of 10 human-annotation tasks. In general, we ask LLMs to perform annotations previously done by humans; we give the LLMs the same instructions as given to humans. The datasets and tasks are selected to represent a diverse range of software engineering research.

Automatic Code Summarization This is an active area, aiming to generate helpful summaries of code [3]. One survey-based study reports that 80% of respondents found code comment generation tools useful [26]. 78% agree that these tools help them understand the source code, especially helping code readability in projects with few comments. Several metrics are used to evaluate code summary quality. All metrics aim to indirectly measure *human perception of quality*, but have limitations. But what criteria are important to evaluate human perception? Haque et al. [5] considered four criteria: accuracy, adequacy, conciseness, and similarity—that should be considered when evaluating code summary quality.

Haque et al. [5] recruited experienced programmers via Upwork, paying them USD60/hr (the applicable market rate). The programmers were presented with 210 functions (from LeClair *et al* [27]) along with associated human-generated, and model-generated summaries (420 in total). Subjects responded to four questions, related to each of the criteria above. Each participant rated a subset of the 420 samples[1], while ensuring that each sample was rated by at least three people.

To elicit LLM responses, we prompt the model with the same guidelines and questions as in the original study. That is, the LLM receives the functions and associated comments, and is tasked to rate its agreement with four statements:

- Independent of other factors, I feel that the summary is accurate.
- The summary contains all the important information, and that can help the understanding of the method.
- The summary contains only necessary information.[2]
- These two comments are similar.

There are four options: 'Strongly agree', 'Agree', 'Disagree', and 'Strongly disagree'. To enhance model performance and maintain a specific output format, we use few-shot learning [17], [18]: we prompt the model with a few illustrative query-response pairs and expect the model to respond to the "test" query of our interest. Prompt-size constraints limited us to three shots for most of the experiments.

Name-Value Inconsistencies Variable names are crucial for code understanding [19]. If a variable name does not match the value stored in it, we have an undesirable *"name-value inconsistency"*. Patra and Pradel [19] user study had 11 participants, and created a dataset evaluating name-value consistency[3]. This dataset includes 40 samples rated by all 11 raters. The raters assigned a Likert score from 1 to 5, where 1 indicates difficulty in understanding (or the presence of name-value inconsistency), and 5 indicates ease of understanding. Suppose there is a variable named `name` to which a float, 2.5, is assigned. This is confusing for developers, since the expected value is a string. All annotators rated it as difficult to understand. On the contrary, another variable `done`, assigned the boolean value False, is easy to understand and was rated as easy to understand by most annotators. We challenge the LLM to perform the same tasks, again with a few-shot prompt.

Causality Causal knowledge supports reasoning about requirements dependencies, particularly for tasks such as test-case construction. Causal relations (e.g., 'If event 1, then event 2') are common in system behavior. However, extracting causality from natural language documents is difficult. Fischbach et al. [29] introduced a dataset, built using 6 raters to develop and evaluate a tool for extracting causal dependencies.[4] The dataset is quite large, with over 10,000 samples. We randomly select 1,000 samples where at least two raters assessed each sentence. The ratings are binary: 1 indicates the presence of

[1]https://github.com/similarityMetrics/similarityMetrics

[2]As noted in Truong *et al* [28], LLMs do not handle negation accurately; so this and the above instruction were slightly modified to remove negation, without changing meaning.

[3]https://github.com/sola-st/Nalin

[4]https://github.com/fischJan/CiRA

a causal relation, and 0 indicates its absence. We elicit such annotations from LLM using a few-shot prompt.

Semantic Similarity Functions are often re-implemented several times, with similar functionality, in software projects. However, semantically-similar functions may be implemented in different ways. Finding and merging such functions could reduce maintenance costs. Kamp et al. [30] applied text similarity measures to JavaDoc comments mined from 11 open-source repositories and then manually classified a selection of 857 pairs to create a realistic dataset.[5] There are three annotation tasks related to functional similarity: goals, operations, and effects. Like for code summarization, each sample was rated by three different people from a pool of eight raters. The original dataset includes some metadata and URLs to the functions but does not contain the function bodies. We retrieve the function bodies and found both bodies for 786 pairs, so we conduct our evaluation on these samples. For each of these pairs, we query LLMs to perform the same three annotation tasks as in the original study.

Static Analysis Warnings Automatic static analysis tools, (*e.g.* FindBugs), can find coding errors, but can raise false alarms. Pruning false alarms and presenting only (or mostly) actionable warnings to developers can be beneficial. A dataset by Kang *et al*. [4] provides human annotations on the task of determining whether a particular code change effectively addresses a static analysis warning. The dataset has 1,306 samples that cover different warning categories.[6] Two annotators assigned each sample one of three labels: open, closed, or unknown. A static analysis warning is considered "closed" if it was reported in a previous software revision but not in a subsequent reference revision, indicating that the problematic code was altered or removed. Conversely, a warning is labeled as "open" if it appears in both the current testing revision and a later reference revision, suggesting that the warning was not actionable or it was ignored by the developers. Lastly, a warning is marked as "unknown" if the file containing the warning was deleted or modified in a (possibly unrelated) way in the reference revision, making it difficult to confirm whether the warning was actionable. We use diff to present the changes made to the repository. Given the extensive metadata, the prompts for this problem are the longest. Due to rate limitations imposed by some models (e.g., Claude and Gemini) and the need to manage costs, we use 200 samples, chosen uniformly at random, for our experiments. For each of these samples, we ask LLMs to perform the annotation task.

Detailed prompts (including guidelines and few-shot samples) for each dataset are provided in the supplementary material.

B. Models under Consideration

We use both closed and open models. Among closed models, we chose GPT-4, Claude-3.5-Sonnet, and Gemini-1.5-Pro, which are all the best models from their respective families or organizations at the time of the work. We also used the GPT-3.5 model. From the open models, we chose Llama3 (70B) and Mixtral (8x22B), which are also the latest models from the open-model family.

C. Research Questions

We seek to study whether LLM prompting can substitute for human annotation labour, for artifact annotation tasks in SE. Specifically, we study how annotator (inter-rater) agreement changes when replacing human annotations with automated LLM annotations. In the first research question, we examine all three categories of inter-rater agreement: human-human, human-model, and model-model. Specifically, if *human-model* inter-rater agreement is similar to that of *human-human* agreement, it may be possible to substitute (some) human ratings with model ratings. We will also examine the inter-model agreement and how it changes across datasets.

RQ 1. When humans are replaced by LLMs to provide answers for "human-rater" questions in software engineering research, what level of agreement is observed between humans and models?

In Section IV-A we discuss the findings of human-human, human-model and model-model inter-rater agreement. Depending on the dataset and annotation task, we find varying levels of agreement. This raises the question of how one might decide whether a particular task is amenable to replacing human effort with an LLM.

RQ 2. How can we determine if LLMs are NOT usable for a specific task?

In contrast to RQ2, where we consider basic feasibility of replacing human effort with a model, the next question focuses on identifying *specific samples* where we can replace one human rating with a model rating. Although a model may not perform well for all samples, selecting samples where models are successful can still significantly reduce human effort.

RQ 3. How can we determine if an answer from an LLM for a specific sample is likely to be in agreement with a human answer?

In case we have determined if and when to delegate parts of a human-subject study to LLMs, the final research question is about the cost benefits of doing so.

RQ 4. How much human annotation effort could be saved without sacrificing inter-rater agreement?

D. Evaluation Methodology

For RQ1: With each task, we record a) number of annotators and b) samples rated by each annotator. Note that an annotator may not have annotated all the samples in the dataset but rather a subset.[7] After identifying the samples annotated by each human, we compute the inter-rater agreement metric Krippendorff's α for each human-human, human-model, and model-model pair. To clarify, if a human (P1) has rated 5

[5]https://github.com/FAU-Inf2/sesame

[6]https://github.com/soarsmu/SA_retrospective/tree/main

[7]For example, Haque *et al* used six annotators, but each sample only received three annotations.

samples {s1, s2, s3, s4, and s5}, and another human (P2) has rated 5 samples {s3, s4, s5, s6, s7}, we consider only {s3, s4, s5} for computing Krippendorff's α for P1 and P2. Other inter-rater agreement metrics exist; *but Krippendorff's α is applicable to any number of annotators*, each assigning one value to one unit of analysis, to incomplete (missing) data, to any number of values available for annotating a variable, and to different kinds of annotations (binary, nominal, ordinal, interval, ratio, etc.). Given the diversity of the datasets we study, this metric allows us to uniformly apply inter-rater agreement to all datasets.

For RQ2: As mentioned earlier, we compute human-human, human-model, and model-model agreement α. Note that model-model agreement is cheap and automated to determine. To answer this question, we examine the model-model agreements and human-model agreement. If these two are correlated, then model-model agreement can help judge the possibility of having high human-model agreement.

For RQ3: To determine whether asking an LLM for a specific sample is a good idea, we use the output probability of a model's answer as a proxy for the model's confidence. We then study the impact of replacing those human answers where an LLM gives the highest-confidence responses with an LLM answer. To assess the impact, we investigate how the inter-rater agreement changes if we replace *one* human annotator with an LLM for a specific fraction of all samples.

For RQ4: We measure the human effort one could save without reducing inter-rater agreement in a statistically significant way. Our unit of *effort* here is the work required, on average, to label one sample. Saving 100% of the effort for one rating corresponds to getting one annotation on each sample in a dataset from an LLM. Note that this saving typically saves at least one, and sometimes multiple human participants. For example, if a problem requires a total of 300 annotations for 100 samples, then saving 100% rating effort means to replace 100 annotations (which otherwise would be annotated by one or more humans).

IV. Results

A. RQ1: Level of Agreement between Humans and Models

We now present the observed level of agreement between humans and models across different datasets.

Datasets	Human to Human Inter-rater agreement		Human to Model Inter-rater agreement		Model to Model Inter-rater agreement	
	Mean	Median	Mean	Median	Mean	Median
Accuracy (Code Summarization)	0.38	0.44	0.48	0.48	0.76	0.78
Adequacy (Code Summarization)	0.40	0.39	0.41	0.38	0.74	0.73
Conciseness (Code Summarization)	0.24	0.26	0.21	0.26	0.74	0.75
Similarity (Code Summarization)	0.64	0.66	0.44	0.43	0.68	0.68
Name-Value Inconsistencies	0.52	0.52	0.49	0.48	0.66	0.67
Causality	0.44	0.49	0.22	0.28	0.39	0.25
Goals (Semantic Similarity)	0.83	0.83	0.77	0.78	0.82	0.81
Operations (Semantic Similarity)	0.74	0.77	0.67	0.68	0.77	0.79
Effects (Semantic Similarity)	0.71	0.73	0.64	0.65	0.69	0.64
Static Analysis Warning	0.80	0.80	0.15	0.23	0.12	0.09

TABLE I: Summary of inter-rater agreements. The three best performing models (Claude, Gemini, and GPT-4) have been used to report the agreement in this table.

Code Summarization Figure 2 shows heatmaps of inter-rater agreement over individual ratings on experimental samples, for human-human, human-model, and model-model pairs for code summarization criteria: *Accuracy, Adequacy, Conciseness*, and *Similarity*. The original study used six human raters. We add ratings from six different models, along with (for comparison) an automated rater which randomly assigned labels to the samples. Figure 2-(a) shows three zones representing three categories of inter-rater agreement, based on available data.[8]

Model-model agreement is high, for all criteria, especially for the three large models (GPT-4, Gemini, and Claude). Table I indicates that the mean Krippendorff's α is 0.68-0.76. Second, we see that human-model and human-human agreements are in similar ranges, 0.24-0.40 and 0.21-0.48 for the first three categories. Because of the way models are trained, we can expect powerful models to tend to reflect the "majority opinion" they learn from the training corpus.

For *similarity*, human-human agreement is higher than human-model agreement. Although smaller models are also useful, they show some bias towards specific answers. For example, Llama-3 consistently chose between "Strongly agree" and "Strongly disagree", ignoring the options "Agree" and "Disagree". Similarly, human rater "P2" was not particularly satisfied with the *conciseness* of the program, resulting in lower inter-rater agreement with all humans and models.

To summarize, for all criteria, we observed generally similar human-human and human-model inter-rater agreements, with a few exceptions.

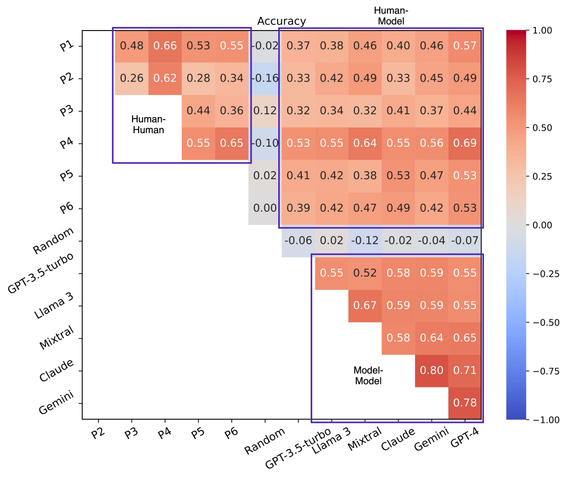
Name-Value Inconsistencies We find that human-model and human-human agreement are quite similar for this task (mean Krippendorff's α 0.49 vs. 0.52). Figure 3 shows a heat-map for all 11 humans, 6 models, and a random rater. We have very high values in all three zones: human-human, human-model, and model-model. Here, too, we find higher agreement between the top 3 models (mean Krippendorff's α 0.66). Note that we will consider only the top three models for reporting Krippendorff's α from here on because we found that only these three models do not show bias or preference towards any specific answer.

Causality The models struggle more to detect causality, relative to the two prior tasks (Figure 4). The mean human-model agreement (0.22) is much less than human-human (0.44). The mean model-model agreement (0.39) is less than 0.5. To summarize, we found lower level of agreement between human-model *and* model-model for this task.

Semantic Similarity For the functional semantic similarity dataset in all categories (e.g., goals, operations, and effects), we have observed high agreement for all human-human (0.71-0.83), human-model (0.64-0.77), and model-model (0.69-0.83) pairs (see Table I and Figure 5). We found the highest level of inter-rater agreement in all three pairings.

Static Analysis Warning For the static analysis warnings rating task, we have only two human raters, and they are in strong

[8]Some human raters did not rate the same samples.


(a) Accuracy

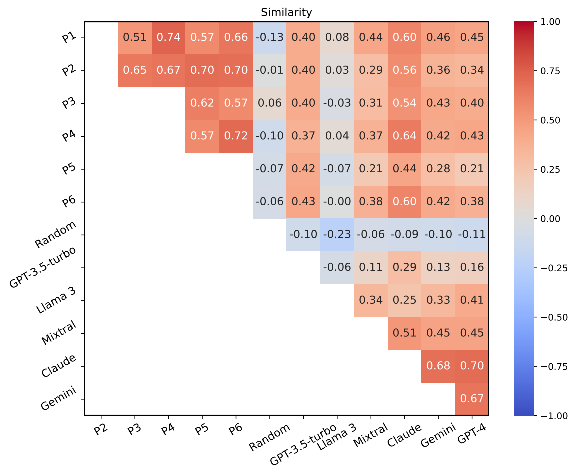

(b) Similarity

Fig. 2: Inter-rater agreement (Krippendorff's α) for code summarization accuracy and similarity. Results for adequacy and conciseness are similar (omitted due to space).

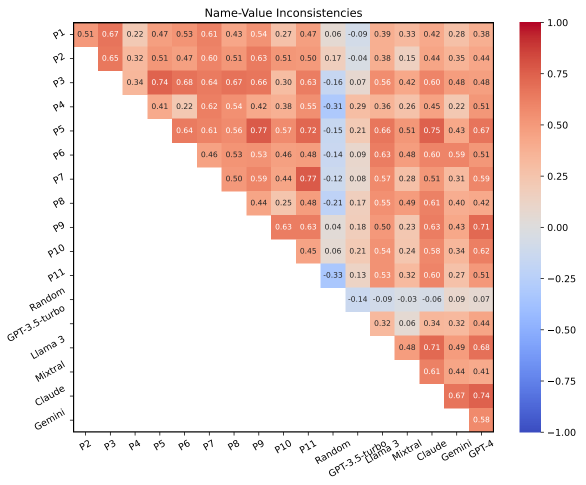


Fig. 3: Inter-rater agreement for name-value inconsistencies.

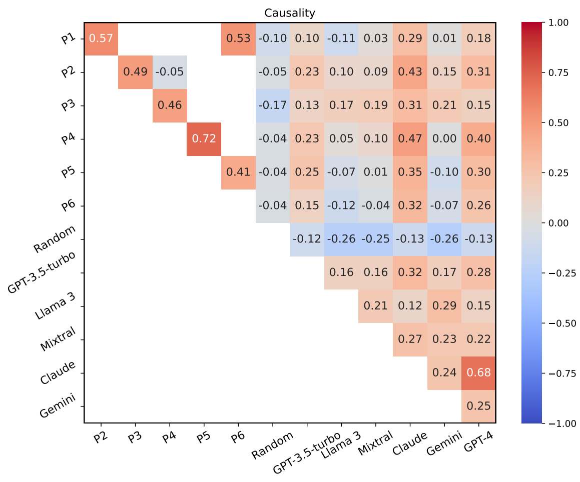


Fig. 4: Inter-rater agreement for causality.

agreement (Krippendorff's α 0.80). However, human-model (0.15) and model-model (0.12) agreements are low (Figure 6).

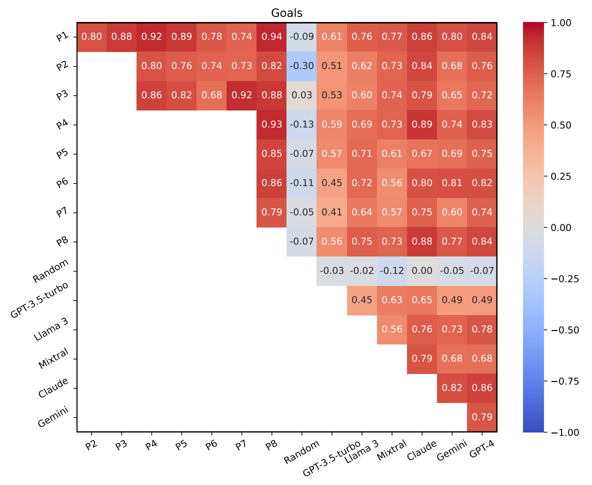


Fig. 5: Inter-rater agreement for semantic similarity with evaluation criteria "Goals". Other criteria ("Operations" and "Effects") give similar heatmaps (omitted due to space).

These findings suggest that LLMs cannot safely substitute for human ratings in this task.

We note that human-model agreement varies from task to task. While it resembles human-human agreement values for some tasks (code summarization, name-value inconsistencies, semantic similarity), it is lower for other tasks (causal relation detection and static analysis warnings). For some tasks, all reviewers labeled all the samples (e.g., static analysis warning and name-value inconsistency). For code summarization, the annotation load was equally distributed among the annotators. In code summarization, each pair of humans had 105 overlapping annotations. However, for causality, the load was not equally distributed. In some cases, there are no samples labeled by two individuals, and in those cases, the corresponding cell is empty.

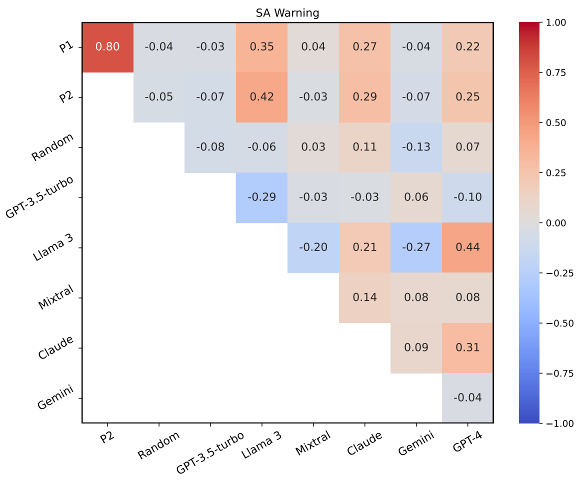


Fig. 6: Inter-rater agreement for static analysis warnings.

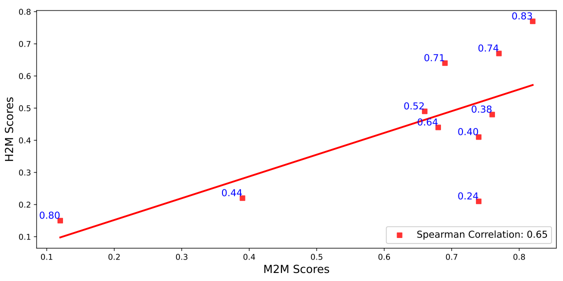


Fig. 7: Human-model inter-rater agreements are positively correlated with model-model inter-rater agreements ($p < 0.05$). Human-human agreements are shown in blue text.

B. *RQ2: LLMs' Applicability for a Specific Task*

We observed that for a majority of tasks, human-model inter-rater agreement is very similar to human-human agreement, while for some tasks, it is lower. What about model-model inter-rater agreement, which does not require any human effort? Figure 7 shows that the mean model-model agreement of the top 3 models is positively correlated with the mean human-model agreement, with a Spearman correlation of 0.65 ($p < 0.05$). This suggests that higher model-model agreement is helpful in indicating good human-model agreement; this observation is potentially valuable in deciding whether to replace humans with models. There is an outlier (high model-model agreement but low human-model agreement) in the bottom-right corner, but for this case (code summarization conciseness), *human-human agreement is also low* (0.24) suggesting that human ratings for this task *per se* are not consistent. Our results suggest that if multiple LLMs reach similar solutions independently, then LLMs are likely suitable for the annotation task.

C. *RQ3: LLMs' Applicability for a Specific Sample*

If we observe the tasks where the models and humans are reasonably agreeing with each other, the model-model agreements are quite high (0.66-0.82). For two datasets, causality detection and static analysis warning, the inter-model agreements are too low (0.39 and 0.12). Now we will see whether we can replace one human rating in each sample with a model. We will discuss the results with respect to the GPT-4 model output from here on, because this model allows us to observe the output probability and is the best-performing model overall.

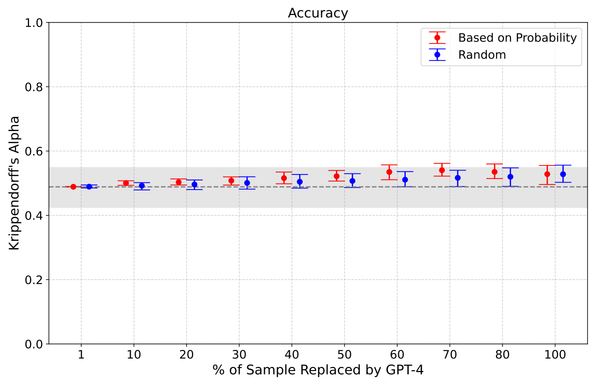
Figure 8(a & b) shows how inter-rater agreement changes with increasing fraction of samples from GPT-4 for code summarization accuracy and similarity. For adequacy and conciseness, we have similar plots to accuracy, so we have omitted them from the paper due to page constraints. Consider Figure 8(a), where we gradually replace the human ratings with model ratings. On the x-axis, 10% means we take 10% of the samples and replace one randomly chosen rating with GPT-4 output, then observe the inter-rater agreement (now including 10% model ratings).

We can choose the 10% samples in two ways: based on GPT-4 output probability or randomly. The red lines show the results when we choose the samples based on probability, and the blue line indicate when samples are chosen randomly. After selecting the samples, we have three ratings for each sample. We randomly select a position, replace the rating with GPT-4 output, and calculate Krippendorff's α as in RQ1. We repeat the process 100 times for each selection criterion. For each data point on the x-axis, we present the mean value and 95% confidence interval. For human-human agreement, we present the inter-rater agreement with a dotted line and have drawn the confidence interval using bootstrapping by randomly selecting 50% of the samples and repeating the process 1,000 times.

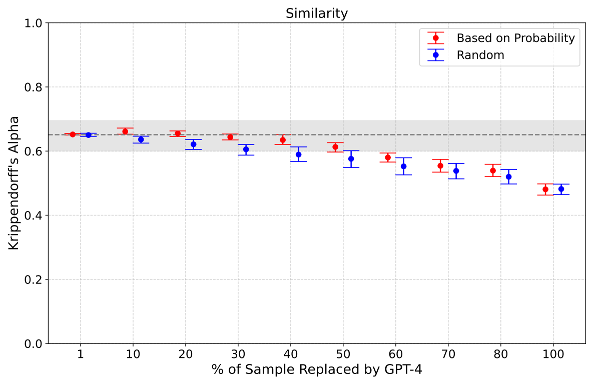
In general, the probability-based selection criterion works better than randomly selected samples. However, at 100%, there is no difference between them because all the samples have been considered for replacement. For the accuracy task of code summarization (Figure 8), we find that the inter-rater agreement does not change at all, even when we replace one rating in all the samples. It slightly increases with the probability-based selection approach, the improvement is insignificant and within the 95% confidence interval of human-human agreement. For the similarity task (Figure 8-b), we have seen that GPT-4 maintains inter-rater agreement up to 50% of the samples while chosen based on probability, potentially saving 16.5% of overall human effort.

For name-value inconsistencies (Figure 9) and functional similarity (figure omitted for space constraints), we observed that we can replace one rating in all samples with GPT-4 and still maintain the same inter-rater agreement. Note that for name-value inconsistency, we have few samples. Therefore, the confidence interval of human-human agreement is much wider.

For causality and static analysis warning, we have low inter-model agreement (0.39 and 0.12). We have already discussed that we should expect low human-model agreement because they are correlated. Figures 10 & 11 show that the inter-rater agreement decreases as we increase the samples from the GPT-4 model, indicating that the model may not be applicable in this setup. However, using the confidence-based approach, we can still see that GPT-4 maintains similar inter-rater agreement,



(a) Accuracy



(b) Similarity

Fig. 8: Inter-rater agreement with increasing % of samples from GPT-4 for code summarization accuracy and similarity. The dotted line indicates the inter-human agreement.

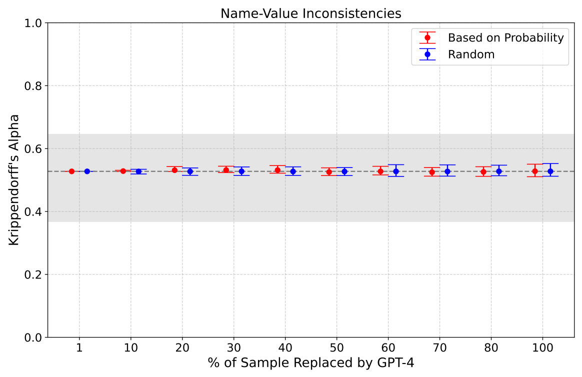


Fig. 9: Inter-rater agreement with increasing % of samples from GPT-4 for name-value inconsistencies.

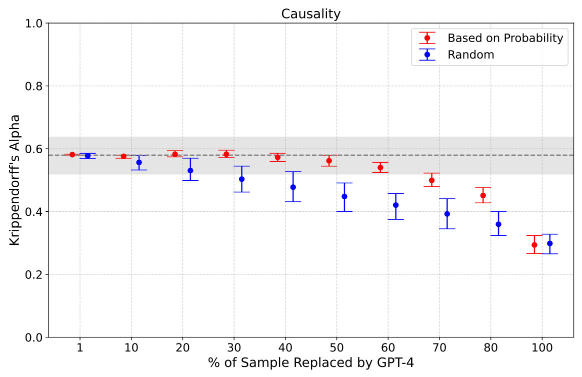


Fig. 10: Inter-rater agreement with increasing % of samples from GPT-4 for causality.

up to 60% for the causality dataset and 50% for the static analysis dataset, showing some promise for using GPT-4 for partial datasets. The confidence interval for these datasets up to that point completely overlaps with the confidence interval of the human-human rating. High-probability samples are likely to be more correct or align better with human preferences compared to other samples.

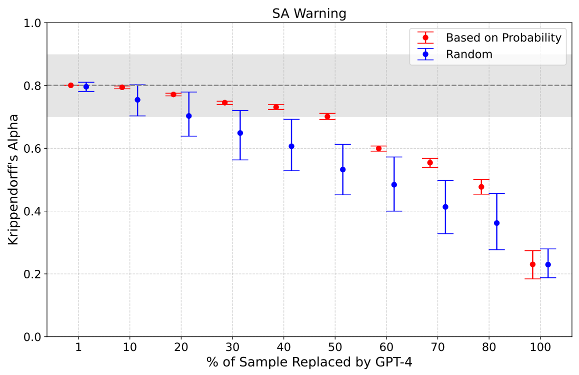


Fig. 11: Inter-rater agreement with increasing % of samples from GPT-4 for static analysis warnings.

Datasets	#of Rating for Each Sample	% Effort Saved for One Rating	% Effort Saved for Overall Process
Accuracy (Code Summarization)	3	100%	33%
Adequacy (Code Summarization)	3	100%	33%
Conciseness (Code Summarization)	3	100%	33%
Similarity (Code Summarization)	3	50%	16.5%
Nalin	11	100%	9%
Causality	2	60%	30%
Goals (Sesame)	3	100%	33%
Operations (Sesame)	3	100%	33%
Effects (Sesame)	3	100%	33%
Static Analysis Warning	2	50%	25%

TABLE II: % of human effort saved for one rating and overall annotation process.

D. RQ4: Human Effort to Potentially Save

By replacing one human rating with an LLM's answer for some fraction of all samples, as studied in RQ3, we can save some human effort. The saved effort depends on two factors: First, the number of ratings required per sample. As we replace only one human's rating for each sample, more required ratings per sample means less potential for saving human effort. Second, the fraction of samples for which we replace a human rating with an LLM's rating. To decide on this fraction, we use the results from RQ3 to determine the maximum fraction that keeps the resulting inter-rater agreement statistically indistinguishable from a human-only study. For example, in Figure 8-b, asking an LLM for help for up to 50% of all samples keeps the agreement within the zone marked with gray background, but going beyond that fraction would change the inter-rater agreement in a statistically significant way.

Based on this reasoning, Table II summarizes the amount of effort that could potentially be saved. For the accuracy task of code summarization, we can save 100% of the effort to obtain one rating per sample, which is 33% of the overall rating effort for this tasks. For name-value inconsistencies, we can save only 9% of the overall effort because there are 11 ratings per sample, whereas for the semantic similarity dataset, we can save up to 33% of the effort because it involves only three ratings per sample. Note that we ignore the few-shot labeling effort (which is 3-4 samples only) for this estimation in Table II. Overall, for seven of the ten tasks, we can safely replace one human rater with a model.

V. Discussion

Deciding Whether and When to Ask an LLM Our findings suggest that we can replace one human rating by a model for a significant number of samples (50%-100%). However, it appears risky to fully replace all humans, even though the probability of the model output at least somewhat indicates the quality of annotations. Based on our findings, we propose the process as given in Figure 12, which involves two steps: 1) Create few-shot examples (3–4 in our setup) to query multiple strong LLMs with all samples, and compute the model-model agreement. 2) If the agreement is high (> 0.5 based on our data), one can safely replace one human rating per sample with an LLM-provided answer. Otherwise, selectively replace one rating only for those samples where the LLM gives a high-confidence answer. Beyond saving human effort, our findings can also help extend a dataset by automatically labeling additional samples. We should emphasize that replacing more than one human can inflate inter-rater agreement because model-model agreements are much higher. This may not fulfill or reflect the original goals of the data annotations.

Confidence Distribution over Samples? We did find that picking the samples based on model confidence yields better results, even when model-model agreement is low, for almost 50% of the data. However, the question remains: what should be our cutoff confidence? Figure 13, which orders fraction of the sample, at decreasing confidence levels, shows that GPT-4 is usually quite confident with its solutions. However, after a certain point, the probability starts to decrease. For causality and static analysis warning, we can see that for the first 50%-60% of samples, the model maintains a high output probability, around 0.80. Thus, the probability cutoff point can be computed when we have low model-model agreement.

Deciding Whether All Humans Are Replaceable In RQ3, we asked if one rating in all samples can be replaced by an LLM. Can we replace *all* humans, but just for *specific* samples? First, we clarify when we can replace all the humans for a sample. Note that there are very few samples where all annotators would agree. For name-value inconsistencies, no sample exists where all annotators agree on a particular rating. Therefore, we will use majority voting to decide whether all humans are replaceable for a sample or not. For a dataset with 2 annotators, the model needs to agree with both annotators, and for a dataset with 11 annotators, the model needs to agree with at least 6 annotators to replace all humans.

Figure 14 shows that samples with higher GPT-4 output probability are more likely to agree with the majority of annotations; however, even with 10% of the samples, there is some possibility of error. Apart from semantic similarity, across all datasets, we can't find any probability split where the model consistently agrees with majority voting. Therefore, we conclude that we cannot replace all the humans for a sample with the current models. We can replace humans for selective samples, but there is some risk associated with it. More study is required for this; we leave it for future research.

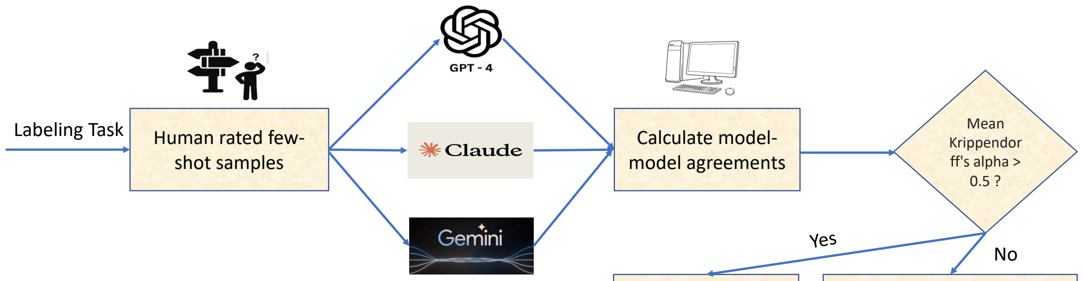
Further Investigations It is now well-established that few-shot learning performs better than zero-shot learning. We also tried zero-shot learning in our preliminary stage and found that the human-model agreement was not satisfactory, even for datasets where few-shot learning performed very well. We tried repeated sampling (at higher temperature): the models mostly generate the same samples. We could improve the multi-sample technique by using chain-of-thought [31] or self-consistency [32], but we do not have the original thoughts used by the original human annotators. In future studies, we strongly recommend collecting human thoughts, which could be used for few-shotting.

VI. Threats to Validity

We consider only 10 human-annotation tasks and 6 models. It is difficult to find publicly available datasets where the ratings from individual annotators are still available; many artifacts contain only the final annotation after resolving disagreements. Even though we see fairly consistent results across the datasets and models, finding may vary with others. Only categorical Krippendorff's α was applicable to all studied datasets. For some of our tasks, ordinal Krippendorff's α is also applicable (e.g., code summarization). However, we observe negligible change from using ordinal Krippendorff's α.

Cohen's κ is another popular measure *but is applicable to only two raters*. We calculated Cohen's κ for each human and model pairs. Although we observe lower values with Cohen's κ, the difference between human-human and human-model agreements remains similar. We share the detailed results in the supplementary material. We proposed a 0.50 inter-model threshold based on our experiments; this may not always work (e.g., in scenarios where typical agreement values are higher or lower). Another possibility (besides using an inter-model agreement threshold), is to manually label a subset of the data; if human-human and human-model IRAs, are similar in this subset, the model could replace one human.

Few-shot learning requires some human effort initially, to get the "shots". However, labeling 3–4 samples is much cheaper compared to labeling all the samples. Also, in few-shot learning, we present the model with one randomly cho-



Labeling Task
Human rated few-shot samples
GPT - 4
Claude
Gemini
Calculate model-model agreements
Mean Krippendorff's alpha > 0.5 ?
Yes
No
Safely replace one rating in each Sample
Selectively replace one rating in some samples based on output probability

Fig. 12: Steps to decide human rating replaceability.

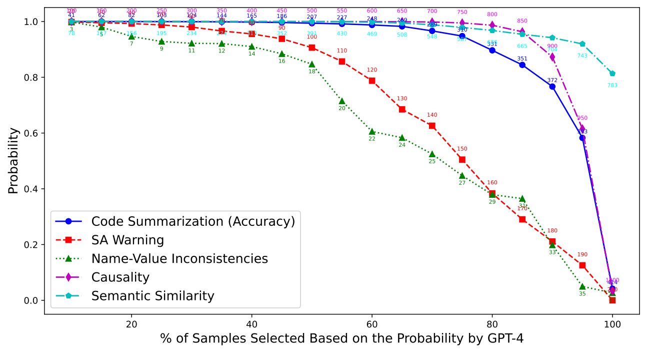


Fig. 13: Change of output probability (by GPT-4) with % of samples sorted based on GPT-4 model's output probability. Some datasets were omitted for clarity of the plot.

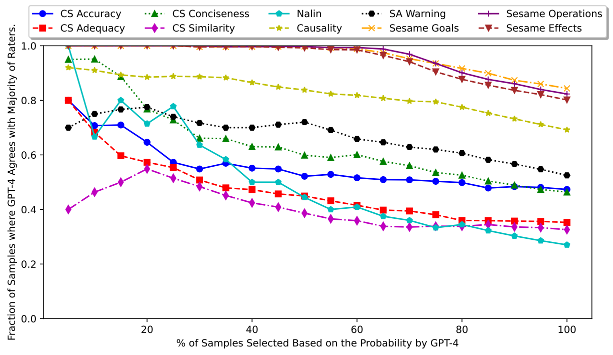


Fig. 14: Change of fraction of samples where GPT-4 agrees with majority voting for % of samples sorted based on GPT-4 model's output probability.

sen example from each category to prevent bias. However, outcomes may depend on the set of few-shot samples used.

Although all of our datasets are relatively recent, some of them might be part of the training data of some of the LLMs we evaluate. As our methodology performs several pre-processing steps, such as function body accumulation (for the semantic similarity task) and diff generation (for the static analysis warnings), it is highly unlikely that the model has memorized the exact results. However, we acknowledge that some bias may exist due to the model's exposure to the underlying GitHub repositories.

VII. Related Work

ML and LLMs in Software Engineering Learning-based approaches and LLMs have been applied to various software development tasks [33], including code completion [34]–[36], test generation [37]–[40], fuzzing [41], [42], automated repair [43]–[47] and coding agents [48]–[51], agents for other software engineering tasks [52], and type prediction [53]–[56]. We follow this trend, but deviate by not targeting a specific task, but by studying the potential for automating human-subject (manual) evaluations, using LLMs.

LLMs to Complement or Replace Human Annotators Recent and concurrent work explores the idea of complementing or replacing human annotators with LLMs in domains beyond software engineering. Work in natural language processing (NLP) compares LLMs and human participants, typically recruited via crowd-working platforms, and report mixed results. While some studies show that adding LLM labels can improve the aggregated labels of a dataset [57] and sometimes outperform crowd-workers [58], others note that LLMs fail to accurately represent differences between demographic groups [59]. Others suggest to use LLMs to perform subtasks in a crowd-sourcing pipeline [60]. Bavaresco et al. [13] propose a benchmark of NLP datasets to compare human and LLM performance. We envision our work to serve as such a benchmark in software engineering. To the best of our knowledge, we are the first to carefully evaluate automated LLM-annotations for software engineering artifacts, and propose actionable guidelines for when and how to use LLMs to create annotations.

Collaborative Human-LLM Annotation Our suggested workflow (Section V) relates to work on human-LLM collaboration for data labeling [61]. Some papers [12], [62] suggest first querying an LLM, and asking humans to refine some of the LLM-provided labels. Instead, we find that inter-model agreement and LLM output probabilities provide an effective way of deciding which labeling tasks can be safely delegated to a model.

LLMs for Data Synthesis and Assessment Beyond using LLMs to partially automate evaluations that would otherwise be performed only by humans, others propose to use LLMs to synthesize additional data to train or fine-tune models for text annotation tasks [63], [64]. There is a recent related survey [65]. A key difference from our work is that training and fine-tuning datasets must be large-scale, but some amount of noise is acceptable, whereas annotations in human-subject studies are typically of smaller scale, but should have high confidence. LLMs have also been proposed as judges of outputs generated by other LLMs, either with a single LLM as the judge [66] or with a set of LLMs that discuss until reaching an agreement [67]. These efforts are primarily about determining human preference, i.e., tasks where different humans may legitimately disagree, whereas many annotation tasks in software engineering have an objectively correct answer that humans can eventually agree upon.

VIII. Conclusion

In this paper, we investigate if LLM responses can substitute for human raters in software engineering annotation tasks, and how such substitutions may affect inter-rater agreement. We find that human-model agreements can be fairly consistent with human-agreements, for the majority of the settings we studied. We also find model-model agreement is a good indication of human-model agreement, suggesting that when powerful models trained on giant human corpora agree with each other, they tend to agree also with humans. Finally, we find that an LLM's confidence (output probability) for a given sample output is a good indication of whether LLM output agrees with majority human rating, for that sample. We caution that the paper considers only discrete (multiple choice) LLM responses, not free-form annotations; furthermore, we have not studied issues of model output bias, nor issues of demographics.

Our scripts and datasets are publicly available: https://zenodo.org/doi/10.5281/zenodo.13146386

Acknowledgments

This work was partially supported by the National Science Foundation under CISE SHF MEDIUM 2107592, the European Research Council (ERC, grant agreements 851895 and 101155832) and by the German Research Foundation within the ConcSys, DeMoCo, and QPTest projects.

References

[1] Y. Zhu and M. Pan, "Automatic code summarization: A systematic literature review," *arXiv preprint arXiv:1909.04352*, 2019.

[2] C. Zhang, J. Wang, Q. Zhou, T. Xu, K. Tang, H. Gui, and F. Liu, "A survey of automatic source code summarization," *Symmetry*, vol. 14, no. 3, p. 471, 2022.

[3] G. Sridhara, E. Hill, D. Muppaneni, L. Pollock, and K. Vijay-Shanker, "Towards automatically generating summary comments for java methods," in *Proceedings of the 25th IEEE/ACM international conference on Automated software engineering*, 2010, pp. 43–52.

[4] H. J. Kang, K. L. Aw, and D. Lo, "Detecting false alarms from automatic static analysis tools: How far are we?" in *Proceedings of the 44th International Conference on Software Engineering*, 2022, pp. 698–709.

[5] S. Haque, Z. Eberhart, A. Bansal, and C. McMillan, "Semantic similarity metrics for evaluating source code summarization," in *Proceedings of the 30th IEEE/ACM International Conference on Program Comprehension*, 2022, pp. 36–47.

[6] D. Roy, S. Fakhoury, and V. Arnaoudova, "Reassessing automatic evaluation metrics for code summarization tasks," in *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2021, pp. 1105–1116.

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

[8] M. Reid, N. Savinov, D. Teplyashin, D. Lepikhin, T. Lillicrap, J.-b. Alayrac, R. Soricut, A. Lazaridou, O. Firat, J. Schrittwieser *et al.*, "Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context," *arXiv preprint arXiv:2403.05530*, 2024.

[9] Anthropic, "Introducing the next generation of claude." [Online]. Available: https://www.anthropic.com/news/claude-3-family

[10] G. V. Aher, R. I. Arriaga, and A. T. Kalai, "Using large language models to simulate multiple humans and replicate human subject studies," in *International Conference on Machine Learning*. PMLR, 2023, pp. 337–371.

[11] R. Futrell, E. Wilcox, T. Morita, P. Qian, M. Ballesteros, and R. Levy, "Neural language models as psycholinguistic subjects: Representations of syntactic state," *arXiv preprint arXiv:1903.03260*, 2019.

[12] A. Goel, A. Gueta, O. Gilon, C. Liu, S. Erell, L. H. Nguyen, X. Hao, B. Jaber, S. Reddy, R. Kartha *et al.*, "Llms accelerate annotation for medical information extraction," in *Machine Learning for Health (ML4H)*. PMLR, 2023, pp. 82–100.

[13] A. Bavaresco, R. Bernardi, L. Bertolazzi, D. Elliott, R. Fernández, A. Gatt, E. Ghaleb, M. Giulianelli, M. Hanna, A. Koller *et al.*, "Llms instead of human judges? a large scale empirical study across 20 nlp evaluation tasks," *arXiv preprint arXiv:2406.18403*, 2024.

[14] O. Dunay, D. Cheng, A. Tait, P. Thakkar, P. C. Rigby, A. Chiu, I. Ahmad, A. Ganesan, C. Maddila, V. Murali *et al.*, "Multi-line ai-assisted code authoring," in *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*, 2024, pp. 150–160.

[15] S. Peng, E. Kalliamvakou, P. Cihon, and M. Demirer, "The impact of ai on developer productivity: Evidence from github copilot," *arXiv preprint arXiv:2302.06590*, 2023.

[16] G. Team, R. Anil, S. Borgeaud, Y. Wu, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth *et al.*, "Gemini: a family of highly capable multimodal models," *arXiv preprint arXiv:2312.11805*, 2023.

[17] T. Ahmed and P. Devanbu, "Few-shot training llms for project-specific code-summarization," in *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, 2022, pp. 1–5.

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

[19] J. Patra and M. Pradel, "Nalin: learning from runtime behavior to find name-value inconsistencies in jupyter notebooks," in *Proceedings of the 44th International Conference on Software Engineering*, 2022, pp. 1469–1481.

[20] S. Easterbrook, J. Singer, M.-A. Storey, and D. Damian, "Selecting empirical methods for software engineering research," *Guide to advanced empirical software engineering*, pp. 285–311, 2008.

[21] J. M. Morse, M. Barrett, M. Mayan, K. Olson, and J. Spiers, "Verification strategies for establishing reliability and validity in qualitative research," *International journal of qualitative methods*, vol. 1, no. 2, pp. 13–22, 2002.

[22] M. Lombard, J. Snyder-Duch, and C. C. Bracken, "Content analysis in mass communication: Assessment and reporting of intercoder reliability," *Human communication research*, vol. 28, no. 4, pp. 587–604, 2002.

[23] J. Cohen, "A coefficient of agreement for nominal scales," *Educational and psychological measurement*, vol. 20, no. 1, pp. 37–46, 1960.

[24] K. Krippendorff, *Content analysis: An introduction to its methodology*. Sage publications, 2018.

[25] L. Finlay, ""outing" the researcher: The provenance, process, and practice of reflexivity," *Qualitative health research*, vol. 12, no. 4, pp. 531–545, 2002.

[26] X. Hu, X. Xia, D. Lo, Z. Wan, Q. Chen, and T. Zimmermann, "Practitioners' expectations on automated code comment generation," in *Proceedings of the 44th International Conference on Software Engineering*, 2022, pp. 1693–1705.

[27] A. LeClair, S. Jiang, and C. McMillan, "A neural model for generating natural language summaries of program subroutines," in *2019*

IEEE/ACM 41st International Conference on Software Engineering (ICSE). IEEE, 2019, pp. 795–806.

[28] T. H. Truong, T. Baldwin, K. Verspoor, and T. Cohn, "Language models are not naysayers: an analysis of language models on negation benchmarks," *arXiv preprint arXiv:2306.08189*, 2023.

[29] J. Fischbach, J. Frattini, A. Spaans, M. Kummeth, A. Vogelsang, D. Mendez, and M. Unterkalmsteiner, "Automatic detection of causality in requirement artifacts: the cira approach," in *Requirements Engineering: Foundation for Software Quality: 27th International Working Conference, REFSQ 2021, Essen, Germany, April 12–15, 2021, Proceedings 27*. Springer, 2021, pp. 19–36.

[30] M. Kamp, P. Kreutzer, and M. Philippsen, "Sesame: A data set of semantically similar java methods," in *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. IEEE, 2019, pp. 529–533.

[31] J. Wei, X. Wang, D. Schuurmans, M. Bosma, F. Xia, E. Chi, Q. V. Le, D. Zhou *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems*, vol. 35, pp. 24 824–24 837, 2022.

[32] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, and D. Zhou, "Self-consistency improves chain of thought reasoning in language models," *arXiv preprint arXiv:2203.11171*, 2022.

[33] M. Pradel and S. Chandra, "Neural software analysis," *Commun. ACM*, vol. 65, no. 1, pp. 86–96, 2022. [Online]. Available: https://doi.org/10.1145/3460348

[34] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. de Oliveira Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H. Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse, A. N. Carr, J. Leike, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever, and W. Zaremba, "Evaluating large language models trained on code," *CoRR*, vol. abs/2107.03374, 2021. [Online]. Available: https://arxiv.org/abs/2107.03374

[35] P. Nie, R. Banerjee, J. J. Li, R. J. Mooney, and M. Gligoric, "Learning deep semantics for test completion," in *ICSE*, 2023.

[36] A. Eghbali and M. Pradel, "De-hallucinator: Iterative grounding for llm-based code completion," *CoRR*, vol. abs/2401.01701, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2401.01701

[37] C. Lemieux, J. P. Inala, S. K. Lahiri, and S. Sen, "Codamosa: Escaping coverage plateaus in test generation with pre-trained large language models," in *45th International Conference on Software Engineering, ser. ICSE*, 2023.

[38] M. Schäfer, S. Nadi, A. Eghbali, and F. Tip, "An empirical evaluation of using large language models for automated unit test generation," *IEEE Transactions on Software Engineering*, 2023.

[39] G. Ryan, S. Jain, M. Shang, S. Wang, X. Ma, M. K. Ramanathan, and B. Ray, "Code-aware prompting: A study of coverage guided test generation in regression setting using llm," in *FSE*, 2024.

[40] J. A. Pizzorno and E. D. Berger, "Coverup: Coverage-guided llm-based test generation," 2024.

[41] Y. Deng, C. S. Xia, H. Peng, C. Yang, and L. Zhang, "Large language models are zero-shot fuzzers: Fuzzing deep-learning libraries via large language models," in *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis, ISSTA*, R. Just and G. Fraser, Eds. ACM, 2023, pp. 423–435. [Online]. Available: https://doi.org/10.1145/3597926.3598067

[42] C. S. Xia, M. Paltenghi, J. L. Tian, M. Pradel, and L. Zhang, "Fuzz4all: Universal fuzzing with large language models," in *ICSE*, 2024.

[43] Z. Chen, S. Kommrusch, M. Tufano, L. Pouchet, D. Poshyvanyk, and M. Monperrus, "SequenceR: Sequence-to-sequence learning for end-to-end program repair," *IEEE Trans. Software Eng.*, vol. 47, no. 9, pp. 1943–1959, 2021. [Online]. Available: https://doi.org/10.1109/TSE.2019.2940179

[44] C. S. Xia and L. Zhang, "Keep the conversation going: Fixing 162 out of 337 bugs for $0.42 each using ChatGPT," 2023.

[45] H. Ye and M. Monperrus, "Iter: Iterative neural repair for multi-location patches," in *ICSE*, 2024.

[46] A. Silva, S. Fang, and M. Monperrus, "Repairllama: Efficient representations and fine-tuned adapters for program repair," 2024.

[47] S. B. Hossain, N. Jiang, Q. Zhou, X. Li, W.-H. Chiang, Y. Lyu, H. Nguyen, and O. Tripp, "A deep dive into large language models for automated bug localization and repair," in *FSE*, 2024.

[48] I. Bouzenia, P. Devanbu, and M. Pradel, "RepairAgent: An autonomous, LLM-based agent for program repair," Preprint, 2024.

[49] J. Yang, C. E. Jimenez, K. Lieret, S. Yao, A. Wettig, K. Narasimhan, and O. Press, "Swe-agent: Agent-computer interfaces enable automated software engineering," 2024.

[50] Y. Zhang, H. Ruan, Z. Fan, and A. Roychoudhury, "Autocoderover: Autonomous program improvement," 2024.

[51] W. Tao, Y. Zhou, W. Zhang, and Y. Cheng, "Magis: Llm-based multi-agent framework for github issue resolution," *arXiv preprint arXiv:2403.17927*, 2024.

[52] I. Bouzenia and M. Pradel, "You name it, i run it: An LLM agent to execute tests of arbitrary projects," 2024. [Online]. Available: https://arxiv.org/abs/2412.10133

[53] V. J. Hellendoorn, C. Bird, E. T. Barr, and M. Allamanis, "Deep learning type inference," in *Proceedings of the 2018 ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/SIGSOFT*, 2018, pp. 152–162. [Online]. Available: https://doi.org/10.1145/3236024.3236051

[54] R. S. Malik, J. Patra, and M. Pradel, "NL2Type: Inferring JavaScript function types from natural language information," in *Proceedings of the 41st International Conference on Software Engineering, ICSE 2019, Montreal, QC, Canada, May 25-31, 2019*, 2019, pp. 304–315. [Online]. Available: https://doi.org/10.1109/ICSE.2019.00045

[55] M. Pradel, G. Gousios, J. Liu, and S. Chandra, "Typewriter: Neural type prediction with search-based validation," in *ESEC/FSE '20: 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Virtual Event, USA, November 8-13, 2020*, 2020, pp. 209–220. [Online]. Available: https://doi.org/10.1145/3368089.3409715

[56] M. Allamanis, E. T. Barr, S. Ducousso, and Z. Gao, "Typilus: neural type hints," in *Proceedings of the 41st ACM SIGPLAN International Conference on Programming Language Design and Implementation, PLDI*, 2020, pp. 91–105. [Online]. Available: https://doi.org/10.1145/3385412.3385997

[57] J. Li, "A comparative study on annotation quality of crowdsourcing and LLM via label aggregation," *CoRR*, vol. abs/2401.09760, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2401.09760

[58] Z. He, C. Huang, C. C. Ding, S. Rohatgi, and T. K. Huang, "If in a crowdsourced data annotation pipeline, a GPT-4," in *Proceedings of the CHI Conference on Human Factors in Computing Systems, CHI 2024, Honolulu, HI, USA, May 11-16, 2024*, F. F. Mueller, P. Kyburz, J. R. Williamson, C. Sas, M. L. Wilson, P. O. T. Dugas, and I. Shklovski, Eds. ACM, 2024, pp. 1040:1–1040:25. [Online]. Available: https://doi.org/10.1145/3613904.3642834

[59] A. Wang, J. Morgenstern, and J. P. Dickerson, "Large language models cannot replace human participants because they cannot portray identity groups," *CoRR*, vol. abs/2402.01908, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2402.01908

[60] T. Wu, H. Zhu, M. Albayrak, A. Axon, A. Bertsch, W. Deng, Z. Ding, B. Guo, S. Gururaja, T. Kuo, J. T. Liang, R. Liu, I. Mandal, J. Milbauer, X. Ni, N. Padmanabhan, S. Ramkumar, A. Sudjianto, J. Taylor, Y. Tseng, P. Vaidos, Z. Wu, W. Wu, and C. Yang, "Llms as workers in human-computational algorithms? replicating crowdsourcing pipelines with llms," *CoRR*, vol. abs/2307.10168, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2307.10168

[61] H. Kim, K. Mitra, R. L. Chen, S. Rahman, and D. Zhang, "Meganno+: A human-llm collaborative annotation system," in *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2024 - System Demonstrations, St. Julians, Malta, March 17-22, 2024*, N. Aletras and O. D. Clercq, Eds. Association for Computational Linguistics, 2024, pp. 168–176. [Online]. Available: https://aclanthology.org/2024.eacl-demo.18

[62] X. Wang, H. Kim, S. Rahman, K. Mitra, and Z. Miao, "Human-llm collaborative annotation through effective verification of LLM labels," in *Proceedings of the CHI Conference on Human Factors in Computing Systems, CHI 2024, Honolulu, HI, USA, May 11-16, 2024*, F. F. Mueller, P. Kyburz, J. R. Williamson, C. Sas, M. L. Wilson, P. O. T. Dugas, and I. Shklovski, Eds. ACM, 2024, pp. 303:1–303:21. [Online]. Available: https://doi.org/10.1145/3613904.3641960

[63] N. Pangakis, S. Wolken, and N. Fasching, "Automated annotation with

generative ai requires validation," *arXiv preprint arXiv:2306.00176*, 2023.

[64] Z. Li, H. Zhu, Z. Lu, and M. Yin, "Synthetic data generation with large language models for text classification: Potential and limitations," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, H. Bouamor, J. Pino, and K. Bali, Eds. Association for Computational Linguistics, 2023, pp. 10 443–10 461. [Online]. Available: https://doi.org/10.18653/v1/2023.emnlp-main.647

[65] Z. Tan, A. Beigi, S. Wang, R. Guo, A. Bhattacharjee, B. Jiang, M. Karami, J. Li, L. Cheng, and H. Liu, "Large language models for data annotation: A survey," *CoRR*, vol. abs/2402.13446, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2402.13446

[66] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. Xing *et al.*, "Judging llm-as-a-judge with mt-bench and chatbot arena," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[67] C. Chan, W. Chen, Y. Su, J. Yu, W. Xue, S. Zhang, J. Fu, and Z. Liu, "Chateval: Towards better llm-based evaluators through multi-agent debate," *CoRR*, vol. abs/2308.07201, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2308.07201