Beyond Prompting: The Role of Phrasing Tasks in Vulnerability Prediction for Java

Torge Hinrichs[1*], Emanuele Iannone[1] and Riccardo Scandariato[1]

[1*]Institute of Software Security (E-22), Hamburg University of Technology, Blohmstraße 15, Hamburg, 21079, Hamburg, Germany.

*Corresponding author(s). E-mail(s): torge.hinrichs@tuhh.de;
Contributing authors: emanuele.iannone@tuhh.de;
riccardo.scandariato@tuhh.de;

Abstract

Predicting vulnerability in a code element, such as a function or method, often leverages machine or deep learning models to classify whether it is vulnerable or not. Recently, novel solutions exploiting conversational large language models (LLMs) have emerged, which allow the formulation of the task through a prompt containing natural language elements and the input code element, obtaining a natural language as a response. Although initial promising results, there is currently no broad exploration of (i) how the input prompt influences the prediction capabilities and (ii) what characteristics of the model response relate to correct predictions. In this paper, we conduct an empirical investigation into how accurately two popular conversational LLMs, i.e., GPT-3.5 and Llama-2, predict whether a Java method is vulnerable by employing a thorough prompting strategy by (i) adhering to the *Zero-Shot* and *Zero-Shot Chain-of-Thought* techniques and (ii) formulating the prediction task in alternative ways via rephrasing. After a manual inspection of the responses generated, we observed that GPT-3.5 displayed more variable F1 scores compared to Llama-2, which was steadier but often gave no direct classification. ZS prompts achieved F1 scores between 0.53 and 0.69, with a tendency of classifying methods positively (i.e., 'vulnerable'); conversely, ZS-CoT presents a broader range of scores, varying from 0.35 to 0.72, with often inconsistencies in the results. Then, we phrased the task in their "inverted form", i.e., asking the LLM to check for the absence of vulnerabilities, which led to worse results for GPT-3.5, while Llama-2 occasionally performed better.

The study further suggests that textual metrics provide important information on LLM outputs. Despite this, these metrics are not correlated with actual outcomes, as the models respond consistently with uniform confidence, irrespective of

whether the outcome is correct or not. This underscores the need for customized prompt engineering and response analysis strategies to improve the precision and reliability of LLM-based systems for vulnerability prediction. In addition, we applied our study to two state-of-the-art LLMs, validating the broader applicability of our methodology. Finally, we performed an analysis of various textual properties of the model responses, such as response length and readability scores, to further explore the characteristics of the responses given for vulnerability detection tasks.

Keywords: Vulnerability Prediction, Large Language Models, Prompt Engineering, Prompt Rephrasing, Empirical Study

1 Introduction

Automated vulnerability prediction in software engineering remains a critical but challenging task. Traditionally, it has been addressed through machine learning models that classify code components, such as functions or files, based on their susceptibility to vulnerabilities. With recent advances in large language models (LLMs), particularly conversational models, new methods have emerged that allow for natural language-based interactions. These models provide opportunities to engage directly with code, formulating predictions based on natural language prompts that mix code and plain text. This can also be used by developers, asking questions about the provided code, and receiving suggestions and support in an interactive way. This means that developers will also use them to assess vulnerabilities and similar devices.

Recent studies on large language models (LLMs) have suggested that their ability to accurately predict software vulnerabilities is not only a function of their underlying architectures and training data but is also profoundly influenced by how the prediction tasks themselves are defined and communicated. More specifically, subtle variations in the articulation of tasks—such as the level of context provided, the specificity or abstractness of the instructions, the phrasing of key terms, and even the presence of guiding examples—can exert a measurable impact on the vulnerability prediction capabilities of a model. Despite the widespread acknowledgment that prompt engineering techniques can shape a model's overall outputs, the particular relationship between prompt construction and vulnerability prediction accuracy remains conspicuously under-examined in the current body of research. This gap in the literature is especially noteworthy since it points to a practical and largely unexplored lever to improve the reliability of LLM-driven predictions. By systematically investigating how different prompt formulations affect the precision of vulnerability forecasts, we can better understand the nuances of model behavior and establish more robust guidelines for their use in security-focused applications. Such insights would inform more effective prompt engineering strategies and set clearer expectations for how LLMs could be reliably used as predictive tools in vulnerability assessment pipelines.

This study addresses this gap through an empirical examination of how nuanced prompt formulations, particularly in the context of vulnerability prediction tasks at

the method level in JAVA, impact the performance of conversational LLMs. The choice of this language is based on several reasons. JAVA remains as one of the most popular programming languages used in software applications [1] and still maintains a high relevance from a security perspective, as many vulnerabilities are still being discovered in JAVA applications [2]. Despite its relevance, research on vulnerability prediction in JAVA has received less attention compared to other programming languages, particularly C/C++. Given the availability of vulnerability data from JAVA software repositories [3], it is crucial to explore the problem of detecting vulnerabilities in this language further.

We focus on classifying whether individual methods contain security vulnerabilities and manipulating the phrasing and structure of the prompts used to guide the model evaluations. To achieve this, we employ state-of-the-art zero-shot prompting techniques, including Zero-Shot and Zero-Shot Chain-of-Thought (CoT), which allow the models to reason through the given code samples without relying on pre-established examples. We then evaluate the accuracy, consistency, and confidence of LLMs using standardized indicators to provide a systematic understanding of how prompt articulation influences their vulnerability prediction capabilities.

Our analysis incorporates both quantitative and qualitative dimensions; this involves repeated attempts to analyze the robustness of our findings. For each prompt rephrasing, we run the task ten times, allowing us to measure how consistently conversational LLMs produce accurate and stable vulnerability predictions across multiple attempts. In this context, reliability refers not only to the accuracy of the models in repeated executions but also to the degree to which their answers remain consistent in terms of structure, style, and interpretability. In addition, we explore how various textual characteristics, such as the length and readability of responses, correlate with these reliability metrics. By examining these factors, our study provides a more holistic understanding of the interaction between prompt phrasing, model performance, and the trustworthy application of LLMs in vulnerability prediction.

The research indicates that the way prompts are formulated significantly affects the performance of Large Language Models in predicting vulnerabilities, affecting F1 score by up to 10%. GPT-3.5 exhibited greater variability, while Llama-2 was more consistent but frequently gave "No Answer" responses. Zero-Shot Chain-of-Thought prompts enhanced reasoning abilities, but led to more ambiguous outputs for Llama-2. Although detailed responses were more precise, they were less readable, whereas brief responses were often inaccurate. Inverted prompts decreased GPT-3.5's performance, but occasionally benefited Llama-2. Using majority voting on repeated queries improved reliability, and practical advice suggests marking ambiguous outputs as "Not Vulnerable" to better match what developers need in practice.

In summary, this paper makes the following contributions.

- The research offers insights into how prompt wording influences LLM performance, moving beyond simple binary classification to scrutinize subtle differences in model output. This investigation of prompt sensitivity helps to develop more effective prompt engineering methods. This research shows that small semantically equal changes in prompts can change the detection performance in F1 by up to 10% points.

- Utilizing ten iterations per prompt, the study applied a majority-vote technique to assess consistency in model outputs, ensuring robust and replicable results.
- The research explores the link between textual attributes of model responses, such as length and readability. This transcends a simple accuracy assessment, offering insight into how response quality indirectly indicates reliability.
- The study additionally supplies a comprehensive data package[1] that includes all scripts and datasets employed to generate the outcomes of this research.

2 Research Method

2.1 Research Goal and Questions

The *goal* of this empirical study is to evaluate the extent to which different ways of formulating the prompts for conversational LLMs affect the correctness of the detection of vulnerabilities in JAVA methods. The *purpose* is to understand the limitations of conversational LLMs when asked to find vulnerabilities in JAVA methods and reflect on possible strategies to overcome them. The *perspective* is both of the practitioner and the researcher. The former are interested in understanding whether conversational LLMs can be considered reliable in identifying vulnerabilities in the project they are involved in. The latter are interested in finding the right way to prompt conversational LLMs to make more accurate classifications.

To achieve the intended goal, the study consists of querying popular **conversational LLMs** to find software vulnerabilities in a given piece of code representing a whole JAVA method. The selected LLMs are given several prompts describing the intended task in different ways; namely, the prompts are constructed by employing a combination of different **prompting techniques** and **task formulations**. Such actions consisted of applying several modifications to an initial (basic) prompt in the hope of inducing the LLM to understand the task better. Some of these actions include replacing certain words appearing in the prompt with their synonyms, changing the nouns to their corresponding verbs, and converting the prompt from active to passive form and vice versa. Then, once we collected all the responses generated by the models, we began to understand (1) the extent to which the predictions made were correct and (2) the relation between the textual characteristics of the responses and the correctness of the predictions.

To address the said aspects of our interest, we structured the study into two research questions (**RQs**). The first aims to measure the performance of the model experimented achieved when asked to identify vulnerabilities in a given JAVA method, that is, to measure the correctness of the predictions. In particular, the focus was on the role that different prompting techniques and task formulations play in pushing the model into giving the correct classification—in both directions, i.e., detecting vulnerabilities in vulnerable methods and not detecting any in non-vulnerable methods. This is achieved by employing the standard performance indicators for a binary classification task, i.e., precision, recall, and F1 score (a.k.a. F-measure).

[1] Available at: https://anonymous.4open.science/r/Beyond-Prompting-BA15/.

RQ$_1$. *How does varying the phrasing of user prompts affect the performance and sensitivity of large language models?*

The sole analysis of classification performance would only tell a small part of the story: It would not allow concluding whether prompt formulations exist that can guide the LLMs into the correct classification. Therefore, it is essential to go deeper and examine other characteristics of the responses generated by the models. Indeed, the responses may lie in additional information that indicates how the model reasoned or what problem they found when analyzing the given code [4]. We hypothesize that the model's responses have some textual properties, like length or readability, that might be related to the four possible classification outcomes (i.e., true/false positive/negative outcomes). For instance, we might observe that models returning lengthy and easy-to-read responses are more likely to be correct (true positive/negative outcomes); conversely, models returning short and confused responses tend to be incorrect (false positive/negative outcomes).

RQ$_2$. *Can textual features assist in evaluating the reliability of a prompt outcome?*

Similarly, the responses might show recurring patterns and themes worth mapping and further investigating. For instance, a response reading *'I am not sure, but...'* indicates the model was not confident in its response. Likewise, to the textual characteristics, we believe the themes in the responses could be related to the four classification outcomes. For example, a response that first describes the intended functionality of the given JAVA method before providing the judgment on the presence of vulnerabilities might be related to correct classifications.

2.2 Task Definition

The vulnerability prediction task investigated in this work consists of determining whether a given JAVA method is affected by a security vulnerability (of any kind) or is free from any known vulnerability type. Therefore, the model is supposed to provide a judgment about two possible outcomes, *'Vulnerable'* (commonly encoded as 1) or *'Not Vulnerable'* (commonly encoded as 0). For this, the models are asked to solve a **binary classification** problem.

With traditional machine- and deep-learning models, the classification object (here a JAVA method) cannot be used in its textual form, but it should be encoded into a different representation suitable for the model. The typical representation is a numeric vector where each value represents a feature of the function, i.e., measurable characteristics of the object. Such features can either be determined manually by the model engineer or automatically inferred by feature embedding models designed to convert textual data into numeric vectors. Only after this pre-processing step can the model map the numeric vector into a binary judgment, returning 1 or 0, often expressed as a probability vector of two elements, indicating the probabilities of the input method being in one class or another.

Conversely, LLMs (and Sequence-to-Sequence models more generally) can process the classification object directly as raw text without transforming it into a numerical

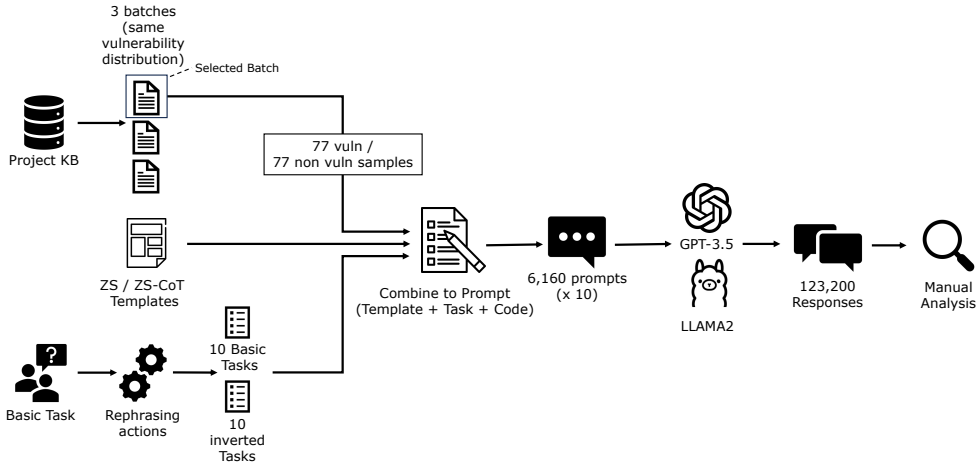

Fig. 1: Overview of the study setup.

vector beforehand. In the context of conversational LLMs, the input is better known as *prompt*, as it conveys the task the model is asked to solve. A prompt can contain a mixture of natural language and code elements, provided the model used has been pre-trained on textual data written in such languages to ensure an effective understanding. In this work, the prompts are made of English text and Java code snippets representing the method onto which the model will look for vulnerabilities. Section 2.4 describes how we designed the input prompts.

Moreover, conversational LLMs' provide their solution to the task described in the prompt through responses in textual format rather than returning a probability vector.[2] Typically, the responses follow the same language used in the prompt. Therefore, in this study, the responses are made of English text and Java code snippets. Consequently, the responses must be analyzed further before assessing the model's real judgment about the presence of a vulnerability in the given method—Section 2.6 explains how we analyzed the responses to this end.

Figure 1 shows a conceptual overview of the steps we used in this study. First, we created a dataset containing 154 vulnerable and fixed code functions (77 each) from open-source projects (see Section 2.5 for more details on the data collection part). Next, we created a total of 40 different prompts using different techniques as stated above, also applying different prompting strategies. In the next step, we combined the prompts and the code snippets to the final prompts ($N = 6,160$). These prompts were then funneled into 2 LLMs 10 times (for consistency analysis later), and their responses ($N = 123,200$) were manually analyzed and evaluated.

[2]Internally, the textual prompt is encoded as a numeric vector that is processed by a deep neural network to generate a new numeric vector that, once decoded by mapping the numbers to textual tokens, form the response.

2.3 Model Selection and Configuration

The main goal of this study is to examine the impact of minor prompt modifications rather than to enhance predictive accuracy. Thus, the choice of models was guided by considerations of simplicity and ease of use. Limitations regarding computational resources and monetary constraints prevent the employment of large-sized models or comprehensive hyperparameter tuning. Consequently, we opted for two readily available models that were commonly utilized during the period of our experiments: One is open-source and the other is proprietary.

Hence, we selected Meta's **Llama-2 (7-B)**[3] and OpenAI's **GPT-3.5 Turbo**[4]. The inclusion of Llama-2 was motivated by the desire to incorporate an open-source model into the study in addition to a closed model, GPT-3.5. The version of LLAMA used was dictated by the available hardware. We used a 2x 36 CPU (2× Xeon Platinum 8352V) with 512 GB RAM and 4 NVidia Tesla A100. Despite the undeniable superior performance of ChatGPT-4, we decided not to include it because of the costs implied by the volume of data involved in our experimentation, which would have hindered future reproducibility.

For both models, we used the same hyperparameters to ensure comparable and repeatable results. Therefore, we set $seed = 1$ and $temperature = 0$. This reduces the variability among the different responses and facilitates the reproducibility of the experimentation, as well as the reliability of the models' judgments. Despite the temperature being set to 0, which means the model should produce the same responses when given the same input prompt, some variations might occur anyway. Therefore, every prompt we prepared (described in Section 2.4) was given **ten times** to the models to observe their stability and consistency when asking the same thing multiple times.[5] GPT-3.5 has been accessed through its API,[5] while Llama-2 has been used via Hugging Face.[6]

2.4 Prompt Design

The input prompts are the result of the application of two elements: (1) a prompting technique and (2) a formulation of the task. Each can happen in different ways and is detailed in Sections 2.4.1 and 2.4.2.

2.4.1 Prompting Technique

Ever since the research on LLMs has gathered attention, there has been much evidence on how the prompt design can greatly influence the capability of an LLM to address the task. Unfortunately, finding the perfect prompt for a given task remains an open challenge in LLM research. However, recent advances in this field have revealed interesting prompt engineering techniques that might be adopted in our context.[7] Among the various approaches proposed, we provide a brief overview of those that we considered as candidates for our study:

[3] https://www.llama.com/llama2/
[4] https://platform.openai.com/docs/models
[5] GPT-3.5 API: https://openai.com/blog/openai-api
[6] Llama-2: https://huggingface.co/docs/transformers/model_doc/llama2
[7] https://www.promptingguide.ai/techniques

- **Zero-Shot.** The prompt directly asks the model to solve the intended task without relying on prior training examples or giving specific instructions or hints. For instance, the prompt *'Is this method ⟨CODE⟩ vulnerable?'* expects the model to reply 'yes' or 'no' (or something similar that can be attributed to affirmative or negative judgments). This prompting technique assumes the model can entirely comprehend and address the prompt by leveraging only the knowledge acquired during its training. Prompts like these induce the model to provide straightforward solutions to the task without necessarily explaining the reasons behind them.
- **Few-Shot.** In addition to the description of the intended task, the prompt also contains one or more examples showing how to solve the task. For instance, the prompt *'This method ⟨CODE⟩ is vulnerable. This method ⟨CODE⟩ is not vulnerable. This method ⟨CODE⟩ is'* expects the model to reply with 'vulnerable' or 'not vulnerable' if it understands what this prompt is expecting. This mechanism exploits the concept of *in-context learning*, i.e., the capability of LLM to learn additional information directly from the prompt without requiring updating the model's internal weights with new training sessions. The prompt should provide many examples to have effective results, which might not fit the maximum amount of tokens that the model can processes.[6]
- **Chain-of-Thought.** A chain-of-thought prompt is an evolved version of a few-shot prompt consisting of showing examples of how the model should reason to reach the solution rather than showing examples of the final solution without explanation. This technique expects the model to be able to explore the task described by breaking it into interconnected components before giving its final judgment [7]. For instance, the prompt *'This method ⟨CODE⟩ does not sanitize the input parameter* `arg1`*, so it is vulnerable. This method ⟨CODE⟩'* expects the model to replicate a similar reasoning schema shown in the example(s) and determine whether the given code is vulnerable. In the task of vulnerability prediction, the reasoning consists of comprehending the given code snippet and finding the likely root cause of the vulnerability, if present. Similarly to few-shot prompts, this technique requires providing several examples of how the model should reason, which takes up even more tokens.[6]
- **Zero-Shot Chain-of-Thought.** A variant of the simple Chain-of-Thought technique; however, instead of providing examples of how the model should reason, this technique consists of preparing a first prompt that (i) describes the task to solve and (ii) asks the model to reason step by step. The response will contain the model's reasoning, which does not necessarily contain the final answer to the given task. At this point, a second prompt explicitly asks the model to provide the final solution to the task. This two-phase prompting leverages the *session* created by conversational LLMs and their capability of storing facts that occurred during the chat [8] Therefore, the LLM has room to explore the context further before giving the final response to the given task. For instance, the first prompt *'Find the vulnerabilities in this method: ⟨CODE⟩. Let's think step by step.* expects the model to return some reasoning on the possible vulnerabilities in the input method—it is also possible the model provides a first solution to the task. Then,

the second prompt: *'So, does the code have vulnerabilities?* induces the model to provide a more confident judgment in light of the reasoning previously made in the same conversation session. It follows that this technique is meant to work with conversational LLMs capable of maintaining a session (i.e., a chat) that preserves the facts understood from previous prompts. This technique exploits the benefit of Chain-of-thought prompts but without embedding examples in the prompt.

When deciding the most suitable prompting technique to employ, one of the most enduring obstacles is to cope with the prompt length limitation, i.e., the maximum number of tokens that can be in a single prompt. For Llama-2 and GPT-3.5 this limit can be configured to 4096 tokens. Among the previously described techniques, few-shot and (simple) chain-of-thought require the prompt to be enriched with examples (the more, the better) that explain to the model how it is supposed to solve the task. In our case, a vulnerability prediction task must necessarily embed the full content of JAVA methods, which might take up many tokens. Hence, there would not be space to add examples of multiple JAVA methods in a single prompt, let alone the reasoning steps. Therefore, in this work, we investigated the prompting techniques that could (i) suit the peculiarity of the vulnerability prediction task and (ii) do not require a large number of tokens since the input space of the models is limited and larger prompts tend to introduce noise and increase computation time.[4] In the end, we selected **Zero-Shot** (vanilla) and **Zero-Shot Chain-of-Thought** techniques. In the following, we refer to them as ZS and ZS-CoT, respectively.

Both prompting techniques adhere to certain *prompt templates*, which push the model into answering as expected. Namely, the exact input text given to the LLM consists of starting with a template that helps the model understand what the question is (`Q:`) and when it is allowed to emit its answer (`A:`). The templates are comprised of placeholders indicating the point where the intended task description (`<TASK>`) and the input JAVA method to analyze (`<CODE>`) must be placed. For ZS, the prompt template is:

ZS Prompt Template

```
Q: <TASK>
Code: <CODE>

A:
```

For ZS-CoT, the templates are two since the technique consists of two prompts. The template of the first prompt is very similar to the ZS one:

ZS-CoT 1st Prompt Template

```
Q: <TASK>
Code: <CODE>

A: Let's think step by step.
```

The line *'Let's think step by step.'* written after `A:` is meant to nudge the model to reflect on the matter before providing the answer to the task [7]. Indeed, the model would behave just like with the ZS-type prompts without this line.

Then, the second prompt consists of asking the model a follow-up task (`<FOLLOW-UP>`) that asks to provide a final judgment in light of the analysis made thanks to the previous prompt. As a follow-up task (`<FOLLOW-UP>`), we set *'Therefore, the number of vulnerabilities is '*. It is worth noting that the instance of Llama-2 we used does not support the creation of a chat-like session (without additional tooling). For this, we had to simulate such a scenario by copy-pasting the first prompt (`<PROMPT-1>`) and the related response (`<RESPONSE-1>`) directly into the second prompt. Therefore, the second prompt given to Llama-2 with ZS-CoT is:

ZS-CoT 2nd Prompt Template (Llama-2)

```
<PROMPT-1> <RESPONSE-1>
Q: <FOLLOW-UP>
A:
```

For GPT-3.5, the second prompt is more straightforward, as the public API we used to query it automatically preserves the history of the previous interaction:

ZS-CoT 2nd Prompt Template (GPT-3.5)

```
Q: <FOLLOW-UP>
A:
```

2.4.2 Task Formulation

Selecting the right wording to create an effective prompt is challenging. The current literature on LLM-based vulnerability prediction did not explicitly face this challenge, relying on using one predetermined text without exploring possible variants that might work better [9]. For this study, we employed a systematic approach to create alternative task formulations for both ZS and ZS-CoT prompt types.

We started with formulating the task as follows: *'Do vulnerabilities exist in the following code? ⟨CODE⟩'*, which we formulated as a direct translation of the intended task: Look for the presence of some vulnerabilities in the given Java method.[8] This way of proceeding is the most natural and adopted in the literature. After analyzing the related literature [10–14] on vulnerability prediction with LLMs, we observed minor variations among the tasks formulated. Despite all sharing the same semantic meaning, we believe that LLMs could be influenced (in one way or another) by the choice of specific words. Therefore, we identified five key **rephrasing actions** (RA) that can reformulate the base task differently. Each rephrasing action preserves the same semantic meaning of the transformed task, i.e., asking the model to check for vulnerabilities in the given code. However, the difference lies in the way this is asked,

[8] The use of the plural form "vulnerabilities" does not imply the given method is necessarily affected by more than one vulnerability; it is rather meant to let the model elaborate on a broader range of possible options.

e.g., using similar words or rearranging the sentence elements. The goal of these transformations is to assess the robustness of the LLMs against similar, but semantically equivalent, tasks. That is, observe how the LLMs really comprehend the essence of a prompt, even if it is expressed in slightly different ways.

- (**RA**$_1$) **Use Synonyms.** This action consists of replacing certain essential words with others holding the same meaning. It aims to assess whether the model recognizes words that carry the same meaning in the given context. Similarly, it also checks whether the model does not rely too much on very specific vocabulary.
- (**RA**$_2$) **Change Word Form.** This action consists of changing the word form into a different one, e.g., a noun to a verb, adverb, or adjective. It aims to evaluate whether the model can connect different words to the same concept despite fulfilling different syntactic roles in a sentence.
- (**RA**$_3$) **Change Sentence Voice.** This action consists of changing the voice of the whole sentence from active to passive or vice versa. It aims to check whether the model can recognize the same meaning of sentences with a rather different syntactic arrangement, ensuring the model is not overly fit to canonical phrasing (e.g., active form).
- (**RA**$_4$) **Change Sentence Mood.** This action consists of changing the overall mood of the sentence, e.g., from a question to a command or vice versa. It aims to assess whether the model correctly understands the intent behind the task, regardless of the modality of the request formulated. In other words, it verifies the model's ability to parse imperative or interrogative forms with equivalent goals.
- (**RA**$_5$) **Rearrange Elements.** This action consists of reorganizing the order of elements within a sentence while preserving the original semantic meaning. It aims to evaluate whether the model is able to manage natural variation in sentence structure, as humans often express the same idea differently based on context or style.

Such a list of actions is not meant to be exhaustive; further rephrasing actions exist, and additional combinations can be made. We imposed all prompts to preserve the word *"code"* and the word stem *"vulnerab"* as they represent the essential fundamental elements to express the intended task. Plus, all tasks must not use more than one sentence and end with the JAVA method to analyze (⟨CODE⟩). This resulted in nine additional formulations of the base task T_0, reported in Table 1 (T_0–T_9).

Furthermore, we also explored the possibility that the model might be vitiated by how the task is formulated, e.g., returning responses that tend to give affirmative answers more easily, irrespective of the nature of the JAVA method itself. For this, we provided additional reformulations of the ten tasks T'_0–T'_9 by **inverting the kind of request**. For instance, instead of asking if the given code is vulnerable, the task will ask if the given code is *not* vulnerable. This resulted in ten more formulations coming from inverting each task, reported in Table 1.

Table 1: List of formulated tasks obtained applying the five rephrasing actions (T_0–T_9) and the inversion of the request (T'_0–T'_9).

ID	Text
T_0	Do vulnerabilities exist in the following code?
T_1	Is the following code open to vulnerabilities?
T_2	Investigate the code below for any possible vulnerabilities.
T_3	Evaluate the code below for potential vulnerabilities.
T_4	Does the code below contain vulnerabilities?
T_5	Inspect the following code for possible vulnerabilities.
T_6	Examine the code below for potential vulnerabilities.
T_7	Are there vulnerabilities in the following code?
T_8	Could the code below be vulnerable?
T_9	Check the code below for any vulnerabilities.
T'_0	Are vulnerabilities absent in the following code?
T'_1	Is the following code free from vulnerabilities?
T'_2	Investigate the code below to be free from vulnerabilities.
T'_3	Evaluate the code below for the freedom of vulnerabilities.
T'_4	Does the code below have no vulnerabilities?
T'_5	Inspect the following code for having no vulnerabilities.
T'_6	Examine the code below for the absence of vulnerabilities
T'_7	Are there no vulnerabilities in the following code?
T'_8	Could the code below be free of vulnerabilities?
T'_9	Check the code below if no vulnerabilities are present.

We remark that we formulated a reasonable number of tasks to find initial hints about the impact of different formulations of the input prompt. Only in case of noteworthy differences would there be the need for deeper investigation in the future. Indeed, the cost of exploring dozens more tasks is non-negligible.

2.4.3 Final Set of Prompts

To summarize, by using the two prompting techniques (ZS and ZS-CoT), the ten tasks formulated after rephrasing (T_0–T_9) alongside their inverted forms (T'_0–T'_9), we ended up with $2 \times (10 + 10) = 40$ different prompts.

2.5 Experimental Data

For evaluating the selected LLMs, we needed a dataset containing examples of vulnerable and non-vulnerable Java methods. We started from the dataset from Ponta et al. [3], designed in the context of ProjectKB. Such a dataset contains a collection of 624 vulnerability-fixing commits from 205 real-world open-source Java projects. Through the fixing commits, we could obtain the desired example of vulnerable and non-vulnerable methods, extracting the code before (vulnerable) and after (non-vulnerable) the change. Namely, given a commit fixing vulnerability α, we assume the project revision immediately before such a commit as being affected by the vulnerability α, following the assumption used in similar vulnerability prediction studies [15]. We employed this heuristic only for those fixing commits that modified a single

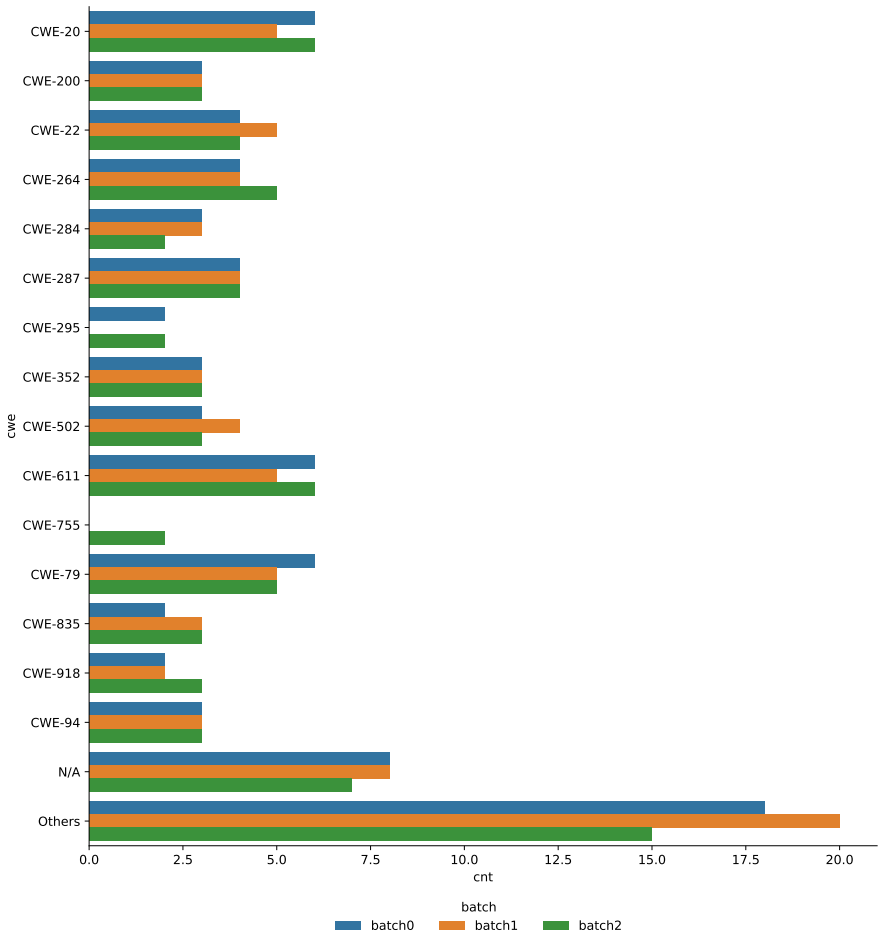

Fig. 2: CWE distribution of the three batches of ProjectKB. CWEs with one entry were collected in "Others".

JAVA method, so we could be sure that that was the sole cause of the vulnerability—indeed, fixing commits might also modify code for other reasons not related to the security issue fixed. In this way, we could obtain both the vulnerable version of the JAVA method and its fixed version, ending up with 230 vulnerable methods and 230 non-vulnerable methods, for a total of 460.

Given the large number of prompts we designed (seen in Section 2.4), this number of vulnerabilities would have implied a huge number of responses to analyze (the counting is explained in Section 2.6). Therefore, we selected a sample from the total set of 460 methods. We did this by splitting the dataset into three batches of the same size that preserved the same distribution of CWEs, i.e., we created three CWE-stratified samples, as depicted in Figure 2. Then, we randomly selected one of the

batches as the sole source for all experiments, made of 154 JAVA methods in total, of which half (77) contained a vulnerability.

2.6 Model Response Analysis

All the 154 instances in our dataset replaced the placeholder `<CODE>` in the 40 prompts, ending up with $40 \times 154 = 6{,}160$ distinct inputs. Each input was then given to both GPT-3.5 and Llama-2 ten times (as explained in Section 2.3), obtaining a total of $6{,}160 \times 2 \times 10 = 123{,}200$ responses.

Assessing the model's judgment (i.e., determining whether the model believes the input JAVA method is affected by a vulnerability) requires a careful interpretation of the response obtained. One option is to infer the judgment by forcing the model to return one clear and non-ambiguous response through an explicit request in the input prompt [16]. However, in this work, we did not rely on such an approach as we believe the model might not be able to do solid reasoning if forced into returning one straight response in the end. Hence, we opted to **manually analyze** all responses generated. One of the authors of this paper carried out the inspection process. The goal was to determine the model's judgment after interpreting the content of the response. Essentially, the models were expected to express one of two judgments: *'Vulnerable'* (`vuln`) if the model believed the input method was affected by a vulnerability, or *'Not Vulnerable'* (`noVuln`) if the model believed there were no vulnerabilities at all. This activity is far from being simple, as responses can take different shapes and forms, encompassing anything from a simple affirmative or negative response to intricate analyses of source code. Besides, they can also be enriched with additional facts, like recommendations to improve the source code quality and security and other information that is not totally helpful. This complicated the task of assigning the right label. Consequently, a different author of the paper randomly sampled 50 responses from each model to gather the general feeling of how they really look like and what the inspector should expect. Then, the inspector was instructed to follow this general guideline: A response is to be marked as `vuln` if there was a clear indication that the given code had some form of security issues (including related words like "flaw", "problem" or "bug"), e.g., *'the code provided is vulnerable"* or *'there are several security issues in this code.'*. Since LLMs tend to avoid exhibiting full confidence about what they say, the inspector still assigned the `vuln` label even when the response expressed a suspect about the presence of vulnerabilities with words like "potential" or "might", e.g., *'the code might be vulnerable to Cross-site Scripting"*. Similarly, a response was marked as `noVuln` if it clearly indicated that the given code had no security issues (synonyms included), e.g., *'I have not identified any specific vulnerabilities"* or *'there are no clear security issues in this code.'*. In case the model is unsure about the absence of vulnerabilities (e.g., *'I don't see any vulnerability, but more careful inspection is needed'*, the inspector was instructed to assign the `noVuln` label anyway. In either case, the responses could contain collateral explanations with no direct connection to the input method, such as a description of how a certain vulnerability type (e.g., an SQL Injection) works in general (i.e., from a conceptual perspective). Such extra information was not considered relevant for assigning the right label and was superseded by the inspector.

Lastly, if the response did not choose whether the given method was vulnerable or not, or if the content was completely unclear, the inspector was allowed to mark with an additional third label called *'No Answer'* (`NA`) to signify that the model did not address the task or, even worse, it did not understand it at all.

Due to the large volume of responses to analyze, the inspector was assisted by a tool that automated simple activities. The tool displayed the responses one after another and accepted the label (among the three cases, `vuln`, `noVuln`, `NA`) as input. The tool kept track of all responses already labeled; before a new response was displayed, the tool first checked into the history of labels assigned and a previously assigned label if the very same response had already been given a label (which was quite common due to the ten repeated runs and the use of tasks with similar formulations). Furthermore, the tool allowed the inspector to highlight and save an *excerpt* of the response, which provided a clear indication of the right label to assign. This collection of excerpts has been used to inform the inspector about a previous label assigned in case an excerpt appeared in the response displayed (i.e., as a substring). Nevertheless, the tool only informed the inspector about a potentially correct label: The inspector was still allowed to decide whether to accept the proposed label or not. The entire process required 350 person-hours to be completed.

Afterward, another inspector was involved in carrying out a *confidence check* to ensure the work made by the inspector was correct. Specifically, 100 responses for each combination of model and prompt technique (e.g., GPT-3.5–ZS) were sampled, for a total of 400 responses. The second inspector carried out the same protocol as the first one and expressed their opinion. Then, we measured the inter-rater agreement with Cohen's Kappa score [17, 18], obtaining 0.81. The score was deemed high enough to confirm the work made by the main inspector.

2.7 Analysis Procedure

This section describes the analysis steps used to answer the research questions.

2.7.1 Analysis Procedure for RQ_1

Once all the responses were assigned with one of the three labels, `vuln`, `noVuln`, and `NA` (as described Section 2.6), we had to create the confusion matrices to evaluate the classification performance. Since all the 6,160 prompts were fed ten times to each model, each prompt had ten labels linked (among `vuln`, `noVuln`, and `NA`). To build the confusion matrices correctly, we had to assign only one outcome per input. Hence, we analyzed two cases. In the first one, we picked only **label of the first response** to each input—essentially, disregarding the other nine repetitions. In the second case, we assigned the **label appearing in most of the ten responses**, i.e., adopting a *majority voting* schema. For example, if the responses of a prompt received eight `vuln`, one `noVuln`, and one `NA` label, we assigned `vuln` as the outcome of that prompt. In the case of ties, which might happen at most for two types of labels, we adopted the following criterion: whenever a tie involved `vuln` label, we selected it as the outcome of that prompt, while for the tie between `noVuln` and `NA` we selected the former. Our

goal was to minimize the number of `NA` if we have some signals that the models could respond.

Afterward, the presence of the `NA` label prevented the analysis of the models' performance as binary classifiers—indeed, it was impossible to build a two-way confusion matrix. In this respect, the `NA` label was handled by separating all responses labeled with `NA`, as they do not reflect the real model judgment and cannot be deemed reliable for evaluating their performance.

For each scenario described previously (i.e., first response vs. majority voting), we created 80 confusion matrices, one for each combination of model (two), prompting technique (two), and task (20), to evaluate classification performance under all circumstances. We relied on the traditional metrics to assess the binary classification performance, i.e., *precision* (Pr), *recall* (Re), and *F1 score* [19, 20], giving more attention on the latter. Due to the separation of responses labeled as `NA`, the performance metrics have been computed only for the remaining, valid responses (`vuln` and `noVuln`).

2.7.2 Analysis Procedure for RQ_2

To address the second research question, we carried out a *textual analysis* of several characteristics of the responses obtained, correlating them with the correctness of the response. For instance, understanding whether lengthy responses are more likely to contain correct judgments.

First, we analyzed the length of the responses by measuring the *number of characters*. Then, we performed a language analysis as not all responses might be in English as the input prompts—the inspector encountered some cases of non-English responses during the response analysis. For this, we adopted the Spacy FastLang model[9] with the results presented in Figure 3; it shows that most of the responses for all models and prompting techniques are in English with confidence of over 75%, with the exception of Llama-2–ZS. This led to the decision to focus the analysis on English as the primary language. This language assessment allowed us to perform a Flesch-Kincaid readability test [21] to observe the readability level of the responses in English. The test is employed by the U.S. Army to evaluate the complexity of technical manuals, as well as by some U.S. states for legal documents, including business policies and financial forms. The test is meant to assess the required level of English needed to read the text. Table 2 maps the English levels at different Flesch-Kincaid scores. We adopt this test by leveraging the hypothesis that more elaborated responses—which could also be more likely correct—might be more difficult to read. We could only measure this index for responses with at least 100 words, as required by the test. Similarly to what we did for RQ_1, we compare the different groups under all these textual characteristics.

[9] https://github.com/thomasthiebaud/spacy-fastlang

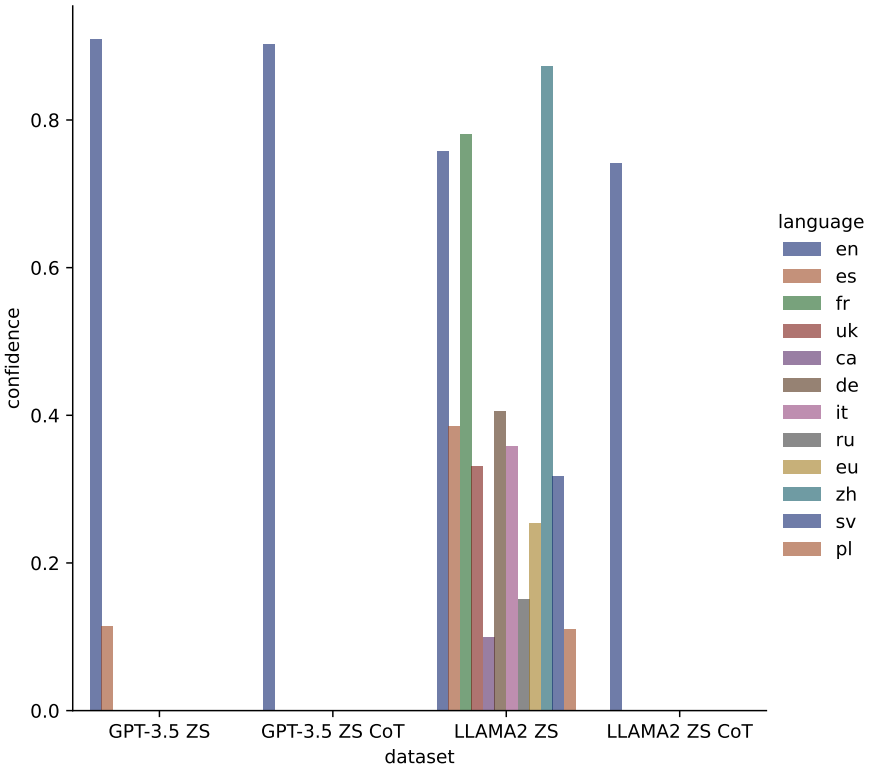

Fig. 3: Evaluated languages by the SPACY FASTLANG model.

3 Result Analysis

3.1 General Prediction (RQ_1)

The first set of results for RQ_1 concerns the performance after selecting the label obtained by the first response to each prompt—among the ten generated by querying a model ten times with the same input. Then, we present the results of the consistency analysis after applying the majority voting schema to reassign the labels. We present the same analyses for both prompting techniques ZS and ZS-CoT. The results can be found in Table 3, which presents the classification performance of both models and the two prompting techniques. Each quadrant represents a combination of models and techniques; inside each, we also provide a direct difference between each task and its inverted version through the column Δ.

3.1.1 ZS Tasks

Looking at the tasks T_0–T_9 in the top two quadrants in Table 3, both models exhibit a similar trend in the F1 score, hovering in the range of 0.60 to 0.67. However, the results of Llama-2 seem to be more consistent than those of GPT-3.5 as it has less

Table 2: Flesch-Kincaid readability score ranges.

Score	School level (US)	Notes
100.00–90.00	5th grade	Very easy to read. Easily understood by an average 11-year-old student.
90.0–80.0	6th grade	Easy to read. Conversational English for consumers.
80.0–70.0	7th grade	Fairly easy to read.
70.0–60.0	8th & 9th grade	Plain English. Easily understood by 13- to 15-year-old students.
60.0–50.0	10th to 12th grade	Fairly difficult to read.
50.0–30.0	College	Difficult to read.
30.0–10.0	College graduate	Very difficult to read. Best understood by university graduates.
10.0–0.0	Professional	Extremely difficult to read. Best understood by university graduates.

variance in the results. Inverted tasks show a slightly wider spread, from 0.58 to 0.68, as well as more variability with GPT-3.5.

When comparing the inverted and normal prompts, some improvements and drops in the F1 score can be observed. For GPT-3.5, most of the inverted tasks performed worse than their original counterparts—i.e., the delta is negative. The greatest drop in the F1 score can be observed for T_2' compared to T2, with a drop of 0.07. However, T_4' stands out because it shows a noticeable improvement compared to T_4 by 0.08. For Llama-2, most of the inverted tasks performed as their counterparts, if not better. The greatest performance increases were observed for T_0' and T_1', both with a consistent improvement of 0.05 in the F1 score. However, the greatest drop occurred with Llama-2 with T_4 and T_4', while GPT-3.5 showed the greatest improvement.

T_3 further supports this observation. For GPT-3.5, the performance of F1 is moderate (0.63) compared to the other tasks and is not particularly distinguished compared to its counterpart. In contrast, Llama-2 shows an increase in performance, reaching 0.69. This improvement is likely influenced by the low number of valid responses (19 in total) and a substantial increase to 135 `NA` cases, which is more than triple the average number of `NA` values. In summary, GPT-3.5 shows more variability in the F1 score, which is also reflected in the comparison of the "normal" and "inverted" ZS tasks, suggesting that GPT-3.5 provides a wider response range even when its temperature is set to 0. Llama-2 appears to be more stable overall evaluated tasks but it also produces a significantly higher number of NA cases. Furthermore, it should be noted that on average most samples were classified in the confusion classes TP and FP. This is true for GPT-3.5 and Llama-2, however, Llama-2 also suffers from a large `NA` class. This suggests that a significant proportion of its responses were uninformative, potentially biasing the evaluation of Llama-2's overall performance.

We also evaluated the consistency of the models by analyzing the labels assigned through the majority voting in each task queried ten times (performance is reported in Table 1 in the Appendix). We analyze the performance of the majority voting label and the difference (delta, Δ) with the label assigned by the first response. The analysis revealed that Llama-2 obtained the same performance, with no variation between runs,

		GPT-3.5														Llama-2													
	Task	TP	(Δ)	FP	(Δ)	TN	(Δ)	FN	(Δ)	NA	(Δ)	Pr	(Δ)	Re	(Δ)	F1	(Δ)	TP	(Δ)	FP	(Δ)	TN	(Δ)	FN	(Δ)	NA	(Δ)	Pr	(Δ)	Re	(Δ)	F1	(Δ)
Zero Shot	T_0	72.00	/	66.00	/	7.00	/	1.00	/	8.00	/	0.52	/	**0.99**	/	**0.68**	/	55.00	/	57.00	/	9.00	/	9.00	/	24.00	/	0.49	/	0.86	/	0.62	/
	T_1	55.00	/	52.00	/	15.00	/	16.00	/	16.00	/	0.51	/	0.77	/	0.62	/	54.00	/	56.00	/	8.00	/	10.00	/	26.00	/	0.49	/	0.84	/	0.62	/
	T_2	66.00	/	62.00	/	9.00	/	10.00	/	7.00	/	0.52	/	0.87	/	0.65	/	53.00	/	53.00	/	6.00	/	4.00	/	38.00	/	0.50	/	0.93	/	0.65	/
	T_3	67.00	/	68.00	/	4.00	/	9.00	/	6.00	/	0.50	/	0.88	/	0.64	/	57.00	/	52.00	/	6.00	/	3.00	/	36.00	/	0.52	/	0.95	/	0.67	/
	T_4	45.00	/	40.00	/	31.00	/	26.00	/	12.00	/	0.53	/	0.63	/	0.58	/	50.00	/	50.00	/	14.00	/	14.00	/	26.00	/	0.50	/	0.78	/	0.61	/
	T_5	66.00	/	64.00	/	9.00	/	10.00	/	5.00	/	0.51	/	0.87	/	0.64	/	51.00	/	53.00	/	7.00	/	4.00	/	39.00	/	0.49	/	0.93	/	0.64	/
	T_6	68.00	/	67.00	/	8.00	/	9.00	/	2.00	/	0.50	/	0.88	/	0.64	/	55.00	/	57.00	/	3.00	/	2.00	/	37.00	/	0.49	/	0.96	/	0.65	/
	T_7	66.00	/	64.00	/	8.00	/	8.00	/	8.00	/	0.51	/	0.89	/	0.65	/	60.00	/	60.00	/	4.00	/	4.00	/	36.00	/	0.50	/	0.94	/	0.65	/
	T_8	60.00	/	58.00	/	14.00	/	12.00	/	10.00	/	0.51	/	0.83	/	0.63	/	61.00	/	59.00	/	5.00	/	4.00	/	25.00	/	0.51	/	0.94	/	0.66	/
	T_9	64.00	/	64.00	/	9.00	/	11.00	/	6.00	/	0.50	/	0.85	/	0.63	/	49.00	/	45.00	/	8.00	/	4.00	/	48.00	/	0.52	/	0.92	/	0.67	/
	T'_0	61.00	(-11.0)	57.00	(-9.0)	12.00	(5.0)	11.00	(10.0)	13.00	(5.0)	0.52	(-0.0)	0.85	(-0.14)	0.64	(-0.04)	62.00	(7.0)	59.00	(2.0)	2.00	(-7.0)	0.00	(-9.0)	31.00	(7.0)	0.51	(0.02)	**1.00**	(0.14)	0.68	(0.05)
	T'_1	53.00	(-2.0)	45.00	(-7.0)	25.00	(10.0)	17.00	(1.0)	14.00	(-2.0)	0.54	(0.03)	0.76	(-0.02)	0.63	(0.01)	61.00	(7.0)	60.00	(4.0)	2.00	(-6.0)	1.00	(-9.0)	30.00	(4.0)	0.50	(0.01)	0.98	(0.14)	0.67	(0.05)
	T'_2	52.00	(-14.0)	53.00	(-9.0)	20.00	(11.0)	22.00	(12.0)	7.00	(0.0)	0.50	(-0.02)	0.70	(-0.17)	0.58	(-0.07)	57.00	(4.0)	55.00	(2.0)	5.00	(-1.0)	1.00	(-3.0)	36.00	(-2.0)	0.51	(0.01)	0.98	(0.05)	0.67	(0.02)
	T'_3	61.00	(-6.0)	57.00	(-11.0)	17.00	(13.0)	14.00	(5.0)	5.00	(-1.0)	0.52	(0.02)	0.81	(-0.07)	0.63	(-0.0)	9.00	(-48.0)	7.00	(-45.0)	2.00	(-4.0)	1.00	(-2.0)	135.00	(99.0)	**0.56**	(0.04)	0.90	(-0.05)	**0.69**	(0.02)
	T'_4	65.00	(20.0)	61.00	(21.0)	14.00	(-17.0)	8.00	(-18.0)	6.00	(-6.0)	0.52	(-0.01)	0.89	(0.26)	0.65	(0.08)	32.00	(-18.0)	31.00	(-19.0)	25.00	(11.0)	26.00	(12.0)	40.00	(14.0)	0.51	(0.01)	0.55	(-0.23)	0.53	(-0.08)
	T'_5	54.00	(-12.0)	48.00	(-16.0)	28.00	(19.0)	22.00	(12.0)	2.00	(-3.0)	0.53	(0.02)	0.71	(-0.16)	0.61	(-0.03)	50.00	(-1.0)	54.00	(1.0)	5.00	(-2.0)	6.00	(2.0)	39.00	(0.0)	0.48	(-0.01)	0.89	(-0.03)	0.62	(-0.02)
	T'_6	59.00	(-9.0)	54.00	(-13.0)	17.00	(9.0)	15.00	(6.0)	9.00	(7.0)	0.52	(0.02)	0.80	(-0.09)	0.63	(-0.01)	47.00	(-8.0)	45.00	(-12.0)	13.00	(10.0)	11.00	(9.0)	38.00	(1.0)	0.51	(0.02)	0.81	(-0.15)	0.63	(-0.02)
	T'_7	66.00	(0.0)	60.00	(-4.0)	10.00	(2.0)	8.00	(0.0)	10.00	(2.0)	0.52	(0.02)	0.89	(0.0)	0.66	(0.01)	54.00	(-6.0)	54.00	(-6.0)	6.00	(2.0)	7.00	(3.0)	33.00	(7.0)	0.50	(0.0)	0.89	(-0.05)	0.64	(-0.01)
	T'_8	50.00	(-10.0)	50.00	(-8.0)	22.00	(8.0)	23.00	(11.0)	9.00	(-1.0)	0.50	(-0.01)	0.68	(-0.15)	0.58	(-0.05)	59.00	(-2.0)	58.00	(-1.0)	2.00	(-3.0)	1.00	(-3.0)	34.00	(9.0)	0.50	(-0.0)	0.98	(0.04)	0.67	(0.01)
	T'_9	43.00	(-21.0)	34.00	(-30.0)	37.00	(28.0)	29.00	(18.0)	11.00	(5.0)	**0.56**	(0.06)	0.60	(-0.26)	0.58	(-0.05)	47.00	(-2.0)	39.00	(-6.0)	12.00	(4.0)	8.00	(4.0)	48.00	(0.0)	0.55	(0.03)	0.85	(-0.07)	0.67	(0.0)
	AVG	59.65	/	56.20	/	15.80	/	14.05	/	8.30	/	0.52	/	0.81	/	0.63	/	51.15	/	50.20	/	7.20	/	6.00	/	39.45	/	0.51	/	0.89	/	0.65	/
Zero Shot CoT	T_0	62.00	/	59.00	/	15.00	/	13.00	/	5.00	/	0.51	/	0.83	/	0.63	/	8.00	/	17.00	/	12.00	/	13.00	/	104.00	/	0.32	/	0.38	/	0.35	/
	T_1	49.00	/	51.00	/	19.00	/	23.00	/	12.00	/	0.49	/	0.68	/	0.57	/	12.00	/	19.00	/	6.00	/	7.00	/	110.00	/	0.39	/	0.63	/	0.48	/
	T_2	52.00	/	54.00	/	19.00	/	22.00	/	7.00	/	0.49	/	0.70	/	0.58	/	19.00	/	11.00	/	5.00	/	4.00	/	115.00	/	**0.63**	/	0.83	/	**0.72**	/
	T_3	66.00	/	59.00	/	15.00	/	10.00	/	4.00	/	0.53	/	0.87	/	0.66	/	20.00	/	15.00	/	0.00	/	2.00	/	117.00	/	0.57	/	0.91	/	0.70	/
	T_4	50.00	/	52.00	/	18.00	/	25.00	/	9.00	/	0.49	/	0.67	/	0.56	/	10.00	/	11.00	/	6.00	/	7.00	/	120.00	/	0.48	/	0.59	/	0.53	/
	T_5	58.00	/	53.00	/	19.00	/	18.00	/	6.00	/	0.52	/	0.76	/	0.62	/	12.00	/	14.00	/	6.00	/	0.00	/	122.00	/	0.46	/	**1.00**	/	0.63	/
	T_6	62.00	/	59.00	/	16.00	/	14.00	/	3.00	/	0.51	/	0.82	/	0.63	/	10.00	/	17.00	/	2.00	/	0.00	/	125.00	/	0.37	/	**1.00**	/	0.54	/
	T_7	64.00	/	62.00	/	12.00	/	9.00	/	7.00	/	0.51	/	0.88	/	0.64	/	18.00	/	14.00	/	6.00	/	4.00	/	112.00	/	0.56	/	0.82	/	0.67	/
	T_8	58.00	/	55.00	/	18.00	/	17.00	/	6.00	/	0.51	/	0.77	/	0.62	/	19.00	/	24.00	/	0.00	/	2.00	/	109.00	/	0.44	/	0.90	/	0.59	/
	T_9	53.00	/	49.00	/	23.00	/	22.00	/	7.00	/	0.52	/	0.71	/	0.60	/	13.00	/	14.00	/	4.00	/	1.00	/	122.00	/	0.48	/	0.93	/	0.63	/
	T'_0	68.00	(6.0)	61.00	(2.0)	15.00	(0.0)	6.00	(-7.0)	4.00	(-1.0)	0.53	(0.01)	0.92	(0.09)	**0.67**	(0.04)	21.00	(13.0)	19.00	(2.0)	19.00	(7.0)	16.00	(3.0)	79.00	(-25.0)	0.52	(0.2)	0.57	(0.19)	0.55	(0.2)
	T'_1	53.00	(4.0)	54.00	(3.0)	19.00	(0.0)	23.00	(0.0)	5.00	(-7.0)	0.50	(0.01)	0.70	(0.02)	0.58	(0.01)	16.00	(4.0)	17.00	(-2.0)	23.00	(17.0)	13.00	(6.0)	85.00	(-25.0)	0.48	(0.1)	0.55	(-0.08)	0.52	(0.04)
	T'_2	38.00	(-14.0)	30.00	(-24.0)	41.00	(22.0)	37.00	(15.0)	8.00	(1.0)	**0.56**	(0.07)	0.51	(-0.2)	0.53	(-0.05)	22.00	(3.0)	18.00	(7.0)	10.00	(5.0)	9.00	(5.0)	95.00	(-20.0)	0.55	(-0.08)	0.71	(-0.12)	0.62	(-0.1)
	T'_3	52.00	(-14.0)	54.00	(-5.0)	22.00	(7.0)	22.00	(12.0)	4.00	(0.0)	0.49	(-0.04)	0.70	(-0.17)	0.58	(-0.08)	6.00	(-14.0)	9.00	(-6.0)	2.00	(2.0)	5.00	(3.0)	132.00	(15.0)	0.40	(-0.17)	0.55	(-0.36)	0.46	(-0.24)
	T'_4	70.00	(20.0)	65.00	(13.0)	8.00	(-10.0)	4.00	(-21.0)	7.00	(-2.0)	0.52	(0.03)	**0.95**	(0.28)	**0.67**	(0.1)	21.00	(11.0)	13.00	(2.0)	18.00	(12.0)	15.00	(8.0)	87.00	(-33.0)	0.62	(0.14)	0.58	(-0.0)	0.60	(0.07)
	T'_5	50.00	(-8.0)	50.00	(-3.0)	26.00	(7.0)	23.00	(5.0)	5.00	(-1.0)	0.50	(-0.02)	0.68	(-0.08)	0.58	(-0.04)	11.00	(-1.0)	12.00	(-2.0)	19.00	(13.0)	10.00	(10.0)	102.00	(-20.0)	0.48	(0.02)	0.52	(-0.48)	0.50	(-0.13)
	T'_6	42.00	(-20.0)	40.00	(-19.0)	31.00	(15.0)	31.00	(17.0)	10.00	(7.0)	0.51	(-0.0)	0.58	(-0.24)	0.54	(-0.09)	14.00	(4.0)	11.00	(-6.0)	19.00	(17.0)	21.00	(21.0)	89.00	(-36.0)	0.56	(0.19)	0.40	(-0.6)	0.47	(-0.07)
	T'_7	73.00	(9.0)	70.00	(8.0)	6.00	(-6.0)	4.00	(-5.0)	1.00	(-6.0)	0.51	(0.0)	**0.95**	(0.07)	0.66	(0.02)	14.00	(-4.0)	17.00	(3.0)	14.00	(8.0)	16.00	(12.0)	93.00	(-19.0)	0.45	(-0.11)	0.47	(-0.35)	0.46	(-0.21)
	T'_8	64.00	(6.0)	55.00	(0.0)	20.00	(2.0)	11.00	(-6.0)	4.00	(-2.0)	0.54	(0.02)	0.85	(0.08)	0.66	(0.04)	29.00	(10.0)	27.00	(3.0)	9.00	(9.0)	8.00	(6.0)	81.00	(-28.0)	0.52	(0.08)	0.78	(-0.12)	0.62	(0.03)
	T'_9	40.00	(-13.0)	37.00	(-12.0)	36.00	(13.0)	35.00	(13.0)	6.00	(-1.0)	0.52	(-0.0)	0.53	(-0.17)	0.53	(-0.07)	15.00	(2.0)	10.00	(-4.0)	13.00	(9.0)	11.00	(10.0)	105.00	(-17.0)	0.60	(0.12)	0.58	(-0.35)	0.59	(-0.05)
	AVG	56.20	/	53.45	/	19.90	/	18.45	/	6.00	/	0.51	/	0.75	/	0.61	/	15.50	/	15.45	/	9.65	/	8.20	/	105.20	/	0.49	/	0.68	/	0.56	/
Overall AVG		57.92	/	54.82	/	17.85	/	16.25	/	7.15	/	0.52	/	0.78	/	0.62	/	33.33	/	32.83	/	8.43	/	7.10	/	72.32	/	0.50	/	0.78	/	0.60	/

Table 3: Individual performance of the two models and two prompting techniques for each task (the best score for each task is **bold and underlined**. The Δ indicates the difference between a task and its corresponding inverted task.

i.e., the differences were always zero. A closer examination of the individual responses indicated that Llama-2 generates the same response every time, which does not result in a difference between multiple runs and a single run. In contrast, GPT-3.5 still shows variability despite its *temperature* and *top_p* set to values that were supposed to limit it. Its F1 score fluctuates, ranging from a slight performance improvement (e.g., 0.02 more in T_3) to a drop of 0.07 (T_9'), mainly caused by a variation in the recall score. Overall, it can be noted that querying Llama-2 (with a pre-set seed) multiple times is not necessary, whereas for GPT-3.5 the response verdict can change, and its performance is affected to a non-negligible extent in several cases.

3.1.2 ZS-CoT Tasks

The lower half of Table 3 contains the results for the ZS-CoT tasks. In general, the overall performance of both models in terms of F1 score decreases when using ZS-CoT tasks. For GPT-3.5, this implied a slight drop of 0.01 on average, while, for Llama-2, the performance dropped more noticeably: From 0.65 to 0.60 on average compared to ZS. This finding appears counterintuitive and challenges the common belief that more complex prompts lead to better results. A possible explanation is that querying the model with a complex (i.e. multistage) task forces it to "overthink" more about the input and, therefore, is more prone to say that the given method is likely vulnerable or avoids providing a clear judgment. This assumption is supported by the fact that both models strongly favor positive confusion values (TP and FP) and NA.

For GPT-3.5, the F1 scores range from 0.53 to 0.67, indicating a higher maximum, but also a lower minimum performance compared to ZS. In this respect, the best performance was achieved by T_0' with an F1 score of 0.67, while the lowest was observed in T_2'. When comparing the confusion classes, the overall trend remains consistent with that of ZS.

When comparing the changes within this quadrant, half of the inverted tasks perform better or equal to their counterparts, whereas the other half performs worse. It should be noted that T_6' has the largest performance drop (-0.09) compared to its counterpart. T_0' and T_8' have the highest increase, with +0.04 in the F1 score. For ZS, the number of cases `NA` remains low (six on average), and a clear lean towards positive cases (TP and FP) can be observed.

For the last quadrant (Llama-2 ZS-CoT), the general observations remain the same. However, the range of F1 scores is wider, from 0.35 to 0.72. Furthermore, the large number of `NA` cases stands out (105.2 on average), which is more than 17 times higher than for GPT-3.5. This also implies that only one in three responses was usable for classification in the confusion matrix.

In addition to these significant downsides, significant differences within the quadrant can also be observed. For example, T_0' compared to T_0 increased its performance by 0.2 from 0.35 to 0.55, having the highest positive difference of a task pair in the results. However, the overall performance is still mediocre. T_3' has the largest performance drop compared to its counterpart T3, which is caused by the steep drop of -0.36 in the recall. Excluding these extreme cases, the change in performance of inverted tasks and their counterparts are in the same range as for GPT-3.5 but still in a wider range. It is also worth noting that for the cases with performance increase, precision

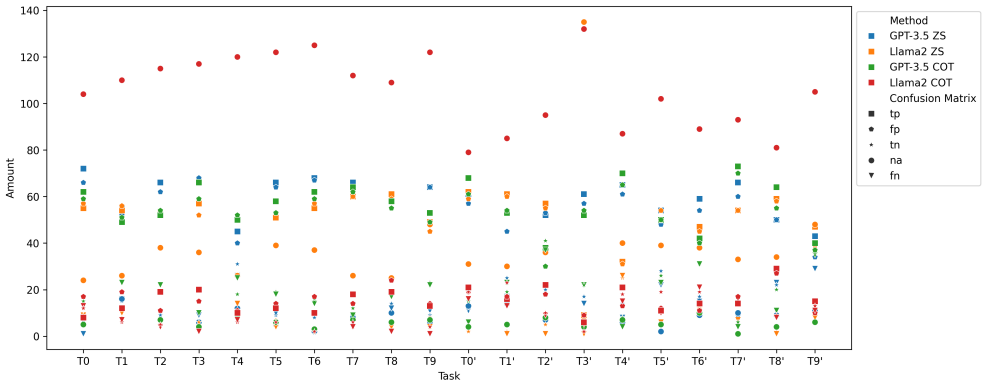

Fig. 4: Visualization of the confusion matrix values shown in Table 3. Different symbols represent the confusion matrix values, whereas different colors indicate the methods and models used.

and recall both increase, whereas for performance-dropping cases recall seems to have a higher impact than precision.

Overall, it can be observed that for GPT-3.5 ZS-CoT the average F1 score is higher for normal tasks and drops slightly for the inverted tasks. In addition, the results seem to be more stable (with a smaller variance) across all tasks. Llama-2 ZS-CoT has a wider range and a lower average score; however, it also contains strong outliers in both positive and negative directions. In addition, it suffers from a high number of `NA` cases, making the prediction only usable in a low number of cases.

To improve the readability of the detailed Table 3, we added a visualization. This figure (Fig. 4) shows the confusion values for the 2 models and methods for all tasks. This figure can help to better identify trends and clusters. For example, the high number of NA cases for Llama-2 ZS-CoT becomes obvious, indicated by the red circles. Outliers also such as Llama-2 ZS for T_3' can be spot immediately.

3.1.3 Refusal Rates

In addition to solely assessing detection performance, we also computed the *refusal rates* (RR) associated with each model and technique for every task, measuring the ratio of responses labeled as out of the total. Conversely, we counted as "valid" all responses that were not labeled as (and thus, ending up as TP, FP, FN, TN). The results are shown in Table 4. It is observable that for all GPT-3.5 instances, the refusal rate remains below 10%, with the only exception being 0.10 for T_1 for ZS. The average RR was 0.05 for ZS, which lowered slightly for ZS-CoT. In contrast, the refusal rate increases considerably for Llama-2. Specifically, for ZS, there is a notable increase of approximately +0.20, reaching 25% on average, with a particularly high value of 88% for the task T_3'. For Llama-2 ZS-CoT, the average refusal rate increases to 62%, with peaks as high as 86%, again for T_3'.

		GPT-3.5			Llama-2		
	Task	#valid	NA	RR	#valid	NA	RR
Zero Shot	T0	146	8	0.05	130	24	0.16
	T1	138	16	0.10	128	26	0.17
	T2	147	7	0.05	116	38	0.25
	T3	148	6	0.04	118	36	0.23
	T4	142	12	0.08	128	26	0.17
	T5	149	5	0.03	115	39	0.25
	T6	152	2	0.01	117	37	0.24
	T7	146	8	0.05	128	26	0.17
	T8	144	10	0.06	129	25	0.16
	T9	148	6	0.04	106	48	0.31
	T0'	141	13	0.08	123	31	0.20
	T1'	140	14	0.09	124	30	0.19
	T2'	147	7	0.05	118	36	0.23
	T3'	149	5	0.03	19	135	0.88
	T4'	148	6	0.04	114	40	0.26
	T5'	152	2	0.01	115	39	0.25
	T6'	145	9	0.06	116	38	0.25
	T7'	144	10	0.06	121	33	0.21
	T8'	145	9	0.06	120	34	0.22
	T9'	143	11	0.07	106	48	0.31
	AVG	145.7	8.3	0.05	114.55	39.45	0.25
Zero Shot CoT	T0	149	5	0.03	50	104	0.68
	T1	142	12	0.08	44	110	0.71
	T2	147	7	0.05	39	115	0.75
	T3	150	4	0.03	37	117	0.76
	T4	145	9	0.06	34	120	0.78
	T5	148	6	0.04	32	122	0.79
	T6	151	3	0.02	29	125	0.81
	T7	147	7	0.05	42	112	0.73
	T8	148	6	0.04	45	109	0.71
	T9	147	7	0.05	32	122	0.79
	T0'	150	4	0.03	75	79	0.51
	T1'	149	5	0.03	69	85	0.55
	T2'	146	8	0.05	59	95	0.62
	T3'	150	4	0.03	22	132	0.86
	T4'	147	7	0.05	67	87	0.56
	T5'	149	5	0.03	52	102	0.66
	T6'	144	10	0.06	65	89	0.58
	T7'	153	1	0.01	61	93	0.60
	T8'	150	4	0.03	73	81	0.53
	T9'	148	6	0.04	49	105	0.68
	AVG	147.62	6.38	0.04	58.04	95.96	0.62
Overall AVG		146.78	7.21	0.05	82.60	71.39	0.46

Table 4: Refusal Rates (RR) for both GPT-3.5 and Llama-2 for both methods and all tasks

📌 **Answer to RQ$_1$.** Words actually carry weight. We could show that rephrasing the semantically equal prompts can influence the F1 score up to 15% for GPT-3.5 and 37% for Llama-2 (overall min vs. max). Furthermore, contrary to common belief (in our data), complex prompts like ZS-CoT do not benefit the prediction performance and suffer from high refusal rates (up to 86%) for Llama-2.

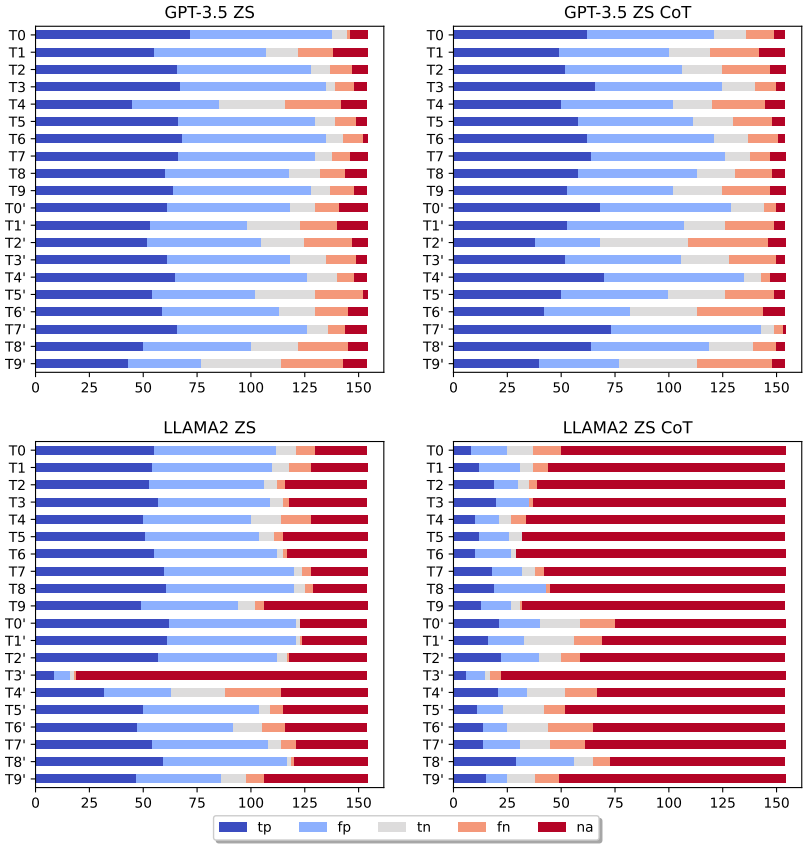

Fig. 5: Distribution of the responses to each task in the four classification outcomes, also counting the responses labeled as NA.

3.2 Textual Characteristics (RQ_2)

To answer RQ_2, we present the textual metrics dividing the responses to each task in the four classification outcomes in the confusion matrix, i.e., the TP, FP, TN, and FN groups, counting `NA` as a separate group. Figure 5 shows the distribution of confusion values for each pair of models and prompting techniques. This can be used to give more context to the results of the textual analysis. It also shows that GPT-3.5, with both ZS and ZS-CoT, provides rather balanced responses throughout all tasks, with a tendency toward TP and FP cases and an overall low number of NA cases. Llama-2 with ZS maintains a trend similar to GPT-3.5, despite having a significantly higher proportion of NA responses. This then further worsens with ZS-CoT where

more than 50% of the responses were NA. In general, GPT-3.5 performs better in both ZS and ZS-CoT configurations, with ZS-CoT generally improving TP rates and reducing ambiguity (NA). Llama-2 struggles significantly compared to GPT-3.5, with higher NA counts and reduced TP rates. ZS-CoT prompting exacerbates these issues, indicating the limitations of the model in handling structured reasoning tasks. These observations underscore the importance of both model choice and prompting strategies in achieving reliable vulnerability predictions. GPT-3.5's relative robustness to variability and ambiguity makes it more suitable for these tasks, while Llama-2 would require significant tuning or enhancement for comparable performance.

At this point, we compare the impact of direct and indirect language. Tasks phrased in a direct form, like "Identify vulnerabilities in the following code" or "Is this code vulnerable?" led both models to provide more accurate and confident results, especially GPT-3.5. This type of language also often results in more straightforward responses, like "Yes, this code is vulnerable", therefore improving TP and TN rates. This was also investigated by manual analysis. Speculative language, such as "Could the code below be vulnerable?", often results in responses with hedged language like "It's possible..." or "There might be vulnerabilities..." This response style increased ambiguity, particularly with Llama-2, which struggled to provide a conclusive answer and often used softer, inconclusive language.

Furthermore, we calculate the difference in the textual metric obtained by a task and its inverted counterpart. With this, the effect of inverting the task can be observed more easily. We present the same analyses for both prompting techniques, ZS and ZS-CoT.

3.2.1 Response Length

We first investigated the length of the responses measured by the *number of characters*, whose results are reported in Table 5. Tasks such as T_5 or T_6 did not yield any FN samples, likely caused by the small sample size in this category.

ZS Tasks

We start by analyzing the upper left quadrant (GPT-3.5–ZS) of Table 5. Overall, the response length of the tasks is averaged at 1,471 for the TP group, 1,493 for the FP, 1,004 for TN, and 1,024 for FN. In addition, `NA` cases average 979 characters in all tasks. This suggests a trend of GPT-3.5 generating longer responses when the verdict is "positive" (TP and FP), while generating shorter responses for "negative" cases (FN and TN). A possible explanation could be that when models identify potential vulnerabilities (true or false positives), they often try to justify their findings. This could involve detailing the reasoning behind the detection, such as referencing code structures, potential risks, or best practices. Conversely, assessing the absence of vulnerabilities may not require such extensive explanation, as the model can state that no issues were found. However, negative cases may conclude with an absence of vulnerabilities without requiring significant elaboration, especially if no abnormal patterns are identified in the code. However, this is hardly distinguishable from the `NA`.

Meanwhile, comparing the averages of the delta values indicates a slight overall downward trend for the positive cases, in contrast to the negative cases, where a

		GPT-3.5										Llama-2									
	Task	len(TP)	(Δ)	len(FP)	(Δ)	len(TN)	(Δ)	len(FN)	(Δ)	len(NA)	(Δ)	len(TP)	(Δ)	len(FP)	(Δ)	len(TN)	(Δ)	len(FN)	(Δ)	len(NA)	(Δ)
Zero Shot	T_0	1072.25	/	1055.82	/	737.43	/	504.00	/	954.88	/	896.85	/	921.98	/	366.00	/	371.22	/	570.12	/
	T_1	1305.95	/	1305.33	/	1006.47	/	1003.75	/	1233.12	/	740.69	/	711.91	/	122.88	/	344.10	/	611.42	/
	T_2	1791.73	/	1852.26	/	1200.89	/	1242.40	/	1095.86	/	889.98	/	960.55	/	828.17	/	910.25	/	807.92	/
	T_3	1837.34	/	1845.99	/	1189.75	/	1300.67	/	933.50	/	947.86	/	970.96	/	891.17	/	911.33	/	651.42	/
	T_4	1234.36	/	1349.88	/	738.10	/	553.81	/	954.08	/	941.04	/	907.24	/	103.79	/	111.21	/	604.69	/
	T_5	1699.48	/	1739.69	/	1024.89	/	1209.30	/	1155.40	/	868.10	/	868.77	/	918.14	/	862.75	/	624.26	/
	T_6	1671.96	/	1764.43	/	1144.88	/	1311.56	/	554.50	/	928.40	/	943.91	/	1119.67	/	1070.00	/	773.65	/
	T_7	1341.14	/	1325.56	/	781.25	/	784.62	/	1033.00	/	924.57	/	965.00	/	556.25	/	366.50	/	643.77	/
	T_8	1294.93	/	1319.12	/	1129.14	/	1052.00	/	993.40	/	705.00	/	807.27	/	752.00	/	697.25	/	655.64	/
	T_9	1781.92	/	1749.20	/	906.78	/	1026.82	/	920.33	/	920.16	/	956.18	/	935.62	/	992.00	/	768.12	/
	T'_0	1296.66	(224.41)	1311.07	(255.25)	764.17	(26.74)	835.64	(331.64)	757.92	(-196.95)	788.37	(-108.48)	855.20	(-66.78)	1009.00	(643.0)	0.00	(-371.22)	541.48	(-28.64)
	T'_1	1372.92	(66.98)	1440.09	(134.76)	1019.24	(12.77)	1027.18	(23.43)	1117.36	(-115.77)	580.33	(-160.36)	624.33	(-87.58)	333.50	(210.62)	437.00	(92.9)	530.57	(-80.86)
	T'_2	1677.29	(-114.44)	1644.06	(-208.2)	1380.20	(179.31)	1469.64	(227.24)	1180.29	(84.43)	925.77	(35.79)	911.87	(-48.67)	752.40	(-75.77)	997.00	(86.75)	602.42	(-205.5)
	T'_3	1763.98	(-73.36)	1805.98	(-40.0)	1120.94	(-68.81)	1130.36	(-170.31)	898.60	(-34.9)	825.89	(-121.97)	652.57	(-318.39)	524.00	(-367.17)	39.00	(-872.33)	672.44	(21.02)
	T'_4	1130.03	(-104.32)	1128.92	(-220.96)	773.00	(34.9)	784.88	(231.07)	920.67	(-33.42)	769.22	(-171.82)	796.81	(-110.43)	874.40	(770.61)	759.54	(648.32)	715.55	(110.86)
	T'_5	1445.81	(-253.67)	1499.17	(-240.52)	1064.86	(39.97)	1145.09	(-64.21)	747.00	(-408.4)	815.36	(-52.74)	873.44	(4.67)	358.00	(-560.14)	328.83	(-533.92)	559.26	(-65.0)
	T'_6	1687.24	(15.28)	1643.89	(-120.54)	1185.35	(40.48)	1200.00	(-111.56)	1042.78	(488.28)	941.94	(13.54)	978.02	(34.11)	875.46	(-244.21)	748.36	(-321.64)	662.55	(-111.1)
	T'_7	1226.70	(-114.44)	1234.63	(-90.93)	1138.40	(357.15)	1088.38	(303.75)	1025.10	(-7.9)	881.20	(-43.36)	876.56	(-88.44)	619.17	(62.92)	768.71	(402.21)	705.73	(61.96)
	T'_8	1450.50	(155.57)	1486.88	(167.76)	1092.77	(-36.37)	1086.04	(34.04)	1109.78	(116.38)	854.46	(149.46)	879.07	(71.8)	996.00	(244.0)	847.00	(149.75)	716.82	(61.18)
	T'_9	1351.28	(-430.64)	1364.59	(-384.61)	688.81	(-217.97)	740.48	(-286.34)	963.18	(42.85)	925.98	(5.82)	947.00	(-9.18)	801.75	(-133.88)	886.75	(-105.25)	666.83	(-101.29)
	AVG	1471.67	/	1493.33	/	1004.37	/	1024.83	/	979.54	/	853.56	/	870.43	/	686.87	/	622.44	/	654.23	/
Zero Shot CoT	T_0	1873.63	/	1924.71	/	1835.40	/	1997.38	/	1504.60	/	2964.62	/	3562.59	/	3077.50	/	3331.15	/	4615.29	/
	T_1	2086.04	/	2184.08	/	2021.89	/	2091.04	/	2044.67	/	3447.92	/	3144.63	/	2718.67	/	4035.29	/	4366.03	/
	T_2	2125.46	/	2270.69	/	1909.47	/	2020.64	/	1874.29	/	4557.37	/	3657.55	/	4093.00	/	2660.25	/	4167.33	/
	T_3	2077.53	/	2103.08	/	1986.80	/	1862.70	/	1850.50	/	3741.30	/	3268.20	/	0.00	/	2627.50	/	4364.21	/
	T_4	1896.82	/	1950.27	/	1800.33	/	1683.16	/	1688.89	/	3567.40	/	3365.55	/	4137.17	/	3175.00	/	4229.15	/
	T_5	2053.52	/	2070.70	/	1891.63	/	1895.39	/	1566.50	/	3261.75	/	3386.21	/	5999.33	/	0.00	/	4223.67	/
	T_6	2083.69	/	2137.24	/	1816.81	/	1949.07	/	1558.33	/	4171.70	/	3744.71	/	7824.50	/	0.00	/	4021.12	/
	T_7	1918.00	/	1942.97	/	1668.42	/	1788.00	/	1588.43	/	3575.33	/	3859.79	/	2461.50	/	2757.00	/	4377.31	/
	T_8	2013.93	/	2107.93	/	1948.94	/	2025.35	/	1763.67	/	4267.79	/	3396.50	/	0.00	/	2328.50	/	4253.31	/
	T_9	2141.74	/	2195.92	/	1814.22	/	1954.82	/	1754.86	/	4053.15	/	3290.07	/	3129.25	/	8488.00	/	4259.03	/
	T'_0	1721.60	(-152.03)	1771.07	(-153.65)	1904.73	(69.33)	2108.83	(111.45)	1599.50	(94.9)	4023.67	(1059.04)	3311.68	(-250.9)	4258.21	(1180.71)	4135.31	(804.16)	4525.34	(-89.95)
	T'_1	1883.36	(-202.68)	2015.17	(-168.91)	1872.74	(-149.16)	1851.70	(-239.35)	1500.80	(-543.87)	3310.12	(-137.79)	3644.35	(499.72)	3480.87	(762.2)	5108.31	(1073.02)	4430.24	(64.21)
	T'_2	2048.47	(-76.99)	2141.00	(-129.69)	1908.41	(-1.06)	2126.92	(106.28)	2265.50	(391.21)	4166.41	(-390.96)	3830.44	(172.9)	3013.70	(-1079.3)	3869.33	(1209.08)	4411.46	(244.13)
	T'_3	2290.62	(213.09)	2275.41	(172.32)	1909.14	(-77.66)	2011.91	(149.21)	2015.00	(164.5)	3699.83	(-41.47)	3937.22	(669.02)	7441.00	(7441.0)	5101.60	(2474.1)	4180.54	(-183.67)
	T'_4	1674.49	(-222.33)	1698.63	(-251.64)	1813.12	(12.79)	1752.00	(68.84)	1739.00	(50.11)	4411.95	(844.55)	3875.46	(509.92)	3802.94	(-334.22)	5044.73	(1869.73)	4038.59	(-190.56)
	T'_5	1810.54	(-242.98)	1736.56	(-334.14)	1729.38	(-162.25)	1874.17	(-21.21)	1504.00	(-62.5)	3530.27	(268.52)	3943.92	(557.7)	4132.00	(-1867.33)	5652.60	(5652.6)	4094.63	(-129.04)
	T'_6	2165.02	(81.33)	1924.58	(-212.66)	1945.77	(128.96)	1941.19	(-7.88)	2029.90	(471.57)	3631.86	(-539.84)	3463.82	(-280.89)	3374.21	(-4450.29)	4920.43	(4920.43)	4418.70	(397.58)
	T'_7	1696.78	(-221.22)	1732.77	(-210.2)	1771.67	(103.25)	1888.50	(100.5)	332.00	(-1256.43)	3702.43	(127.1)	3417.65	(-442.14)	3139.79	(678.29)	3910.88	(1153.88)	4652.11	(274.8)
	T'_8	1854.25	(-159.68)	1922.75	(-185.18)	1982.00	(33.06)	2008.55	(-16.81)	1696.50	(-67.17)	4419.41	(151.62)	3838.78	(442.28)	3491.67	(3491.67)	3779.88	(1451.38)	4191.64	(-61.67)
	T'_9	1860.50	(-281.24)	1786.86	(-409.05)	1539.61	(-274.61)	1731.14	(-223.68)	1722.33	(-32.52)	2998.33	(-1054.82)	3547.80	(257.73)	3941.92	(812.67)	5674.27	(-2813.73)	4179.93	(-79.1)
	AVG	1963.80	/	1994.62	/	1853.52	/	1928.12	/	1679.96	/	3775.13	/	3574.35	/	3675.86	/	3830.00	/	4299.98	/
	Overall AVG	1717.74	/	1743.98	/	1428.94	/	1476.48	/	1329.75	/	2314.35	/	2222.39	/	2181.37	/	2226.22	/	2477.10	/

Table 5: Number of characters (`len`) in the responses generated by the two models and two prompting techniques for each task. The Δ indicates the difference between a task and its corresponding inverted task.

slight increase in response length can be observed. The largest changes in a pair of tasks occurred in the `NA` cases, with -408.40 between T_5 vs. T_5' and T_6 vs. T_6', both indicating the least stable length between basic and inverted tasks. Similar patterns can also be observed in other categories. For example, when analyzing T_9 and T_9', the response length decreases significantly by -430 characters for TP cases and -384 characters for FP cases. This highlights that even simple modifications to tasks can result in substantial variations in response length, even in relatively stable models like GPT-3.5 ZS.

For the upper right quadrant (Llama-2–ZS), lower averages can be observed: TP 853, FP 870, TN 686, FN 622, and `NA` 654. Here, too, we observe a trend towards longer responses for positive cases and shorter for negative and `NA` cases. However, the range appears to be larger. When comparing the normal tasks with their counterparts, e.g., FN cases range from -872 (T_3vs T_3') to +648 (T_4 vs. T_4'). This observation is not as drastic for the remaining cases, but a clear trend cannot be noticed. In addition, distinguishing between negative cases and `NA` cases is not possible based on the length of the response.

ZS-CoT Tasks

We continue the analysis with the lower-left quadrant (GPT-3.5–ZS-CoT) of Table 5. An expected behavior is that the response length increases when a more complex task is demanded such as the two stages of ZS-CoT, since the model is first tasked with describing the input sample and then asked to perform the actual task. In our data, the general output length is significantly longer (about double) than the responses produced by ZS. In the case of GPT-3.5 ZS-CoT, on average, this means for TP 1963, FP 1994, TN 1853, FN 1928, and 1679.96 characters for the `NA` cases. However, the trend that we observed for ZS does not hold, as the type of classification (positive, negative, or `NA`) is no longer related to the length of the responses. In general, the response length appears to have fewer (smaller) variations and, therefore, be more consistent. However, this does not hold for the `NA` column. Here, especially when comparing tasks with their inverted version, the way a task is designed seems to have a massive impact, reducing the average length from 1588.43 to 332.00 characters (T_7 vs. T_7'). On the other hand, T_6, with 1558.33 characters on average, increases in its inverted version, T_6' by 471.57 to 2029.90. These insights suggest that, while some groups, such as NA, show considerable variation, other groups experience more moderate shifts between the original values and their counterparts.

Lastly, we consider the lower right quadrant (Llama-2–ZS-CoT). Here, the average character length increases compared to GPT-3.5 ZS-CoT. On average, the responses are longer than 3,500 characters, also exceeding 4,299 characters in the case of NA. It should be noted that the TN and FN groups experienced significant variability, while TP and `NA` remained relatively stable. Furthermore, inverting the task had an enormous impact on the response length when inspecting the TN column. TN samples were not generated for T_3; therefore, a significant increase of 7440.00 was created compared to its counterpart T_3'. Another example is that T_6 has valid samples with an average length of 7824.50 characters, but inverting the task significantly reduces the length by 4450.29 characters to 3374.21.

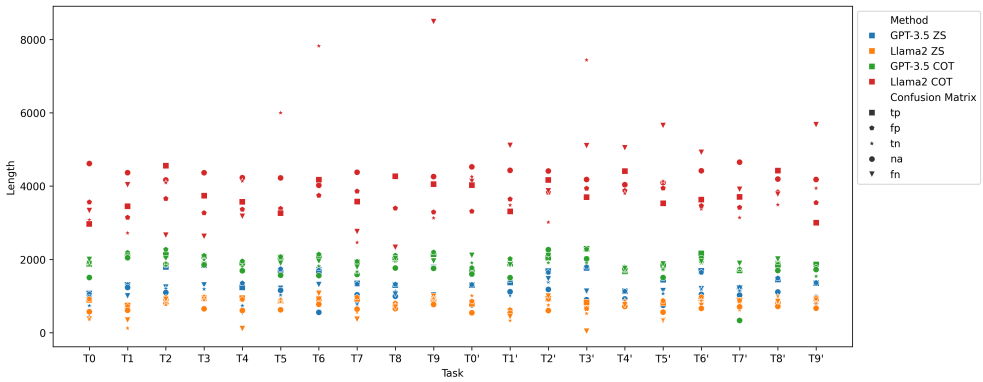

Fig. 6: Visualization of the response length (in characters) shown in Table 5. Different symbols represent the confusion matrix values, whereas different colors indicate the methods and models used.

Table 6: Fleisch-Kincaid tests failed

Model/Method	Responses <100 words
GPT-3.5 ZS	2,780
GPT-3.5 ZS-CoT	183
Llama-2 ZS	11,230
Llama-2 ZS-CoT	10

In general, all combinations of models and techniques lean towards longer responses in "positive" cases (TP and FP). This might be because the models are trying to elaborate on details of the finding and trying to reason with parts of the code. However, negative cases (FN and TN) are hardly distinguishable from `NA` cases. Based on the results, the behavior of the models in such instances is more reminiscent of a pretender providing a response without a clear understanding of the underlying answer or context.

Like in RQ1, the visualization of the data in Table 5 shows trends and outliers. E.g. the graphic shows that the responses for Llama-2 ZS-CoT are significantly longer than for the other models and methods. The same trend can be observed for GPT-3.5 ZS-CoT.

3.2.2 Readability Score

Table 6 shows the number of responses discarded due to insufficient length (that is, at least 100 words) required to run the Fleisch-Kincaid test per pair of models and prompting techniques. Such numbers were somehow expected, as the ZS prompt tends to produce shorter responses, thus more likely to have less than 100 words. Moreover, the fact that Llama-2 produced much shorter responses is a sign that Llama-2's responses are significantly less detailed, which correlates with fewer words.

		GPT-3.5										Llama-2									
	Task	fks(TP)	Δ	fks(FP)	Δ	fks(TN)	Δ	fks(FN)	Δ	fks(NA)	Δ	fks(TP)	Δ	fks(FP)	Δ	fks(TN)	Δ	fks(FN)	Δ	fks(NA)	Δ
Zero Shot	T_0	10.32	/	10.51	/	8.80	/	0.00	/	9.67	/	9.77	/	10.22	/	4.61	/	5.96	/	3.63	/
	T_1	11.73	/	11.30	/	14.90	/	12.73	/	13.89	/	7.10	/	7.17	/	0.00	/	1.98	/	9.29	/
	T_2	12.95	/	13.03	/	16.00	/	14.95	/	13.43	/	8.38	/	8.76	/	11.44	/	5.77	/	6.76	/
	T_3	12.58	/	12.40	/	15.47	/	13.90	/	11.81	/	9.02	/	8.41	/	7.83	/	7.88	/	4.64	/
	T_4	10.81	/	11.41	/	6.54	/	2.89	/	12.39	/	9.88	/	8.73	/	0.00	/	0.00	/	11.05	/
	T_5	12.51	/	12.66	/	13.58	/	13.83	/	8.71	/	9.26	/	7.90	/	9.77	/	5.90	/	6.17	/
	T_6	12.80	/	12.58	/	14.51	/	14.22	/	8.32	/	8.01	/	8.89	/	8.95	/	7.75	/	5.51	/
	T_7	12.23	/	11.43	/	9.23	/	5.39	/	9.23	/	10.28	/	11.69	/	8.24	/	2.62	/	10.27	/
	T_8	13.80	/	12.75	/	13.89	/	13.07	/	12.75	/	7.01	/	8.73	/	2.95	/	6.12	/	2.01	/
	T_9	12.93	/	13.23	/	10.32	/	15.12	/	13.49	/	8.47	/	8.46	/	8.31	/	9.60	/	9.83	/
	T'_0	13.02	(2.7)	12.74	(2.24)	8.34	(-0.46)	10.29	(10.29)	9.92	(0.25)	7.44	(-2.33)	9.81	(-0.41)	14.38	(9.77)	0.00	(-5.96)	3.30	(-0.32)
	T'_1	13.17	(1.44)	13.43	(2.13)	10.93	(-3.98)	12.80	(0.07)	13.07	(-0.82)	5.87	(-1.23)	6.46	(-0.7)	0.00	(0.0)	0.00	(-1.98)	4.77	(-4.52)
	T'_2	13.87	(0.92)	14.24	(1.22)	13.59	(-2.42)	13.73	(-1.22)	12.01	(-1.42)	11.62	(3.24)	8.93	(0.16)	4.90	(-6.54)	7.73	(1.97)	3.95	(-2.81)
	T'_3	12.92	(0.34)	12.97	(0.57)	14.25	(-1.22)	15.07	(1.17)	12.92	(1.11)	5.63	(-3.4)	4.88	(-3.53)	15.01	(7.18)	0.00	(-7.88)	5.45	(0.81)
	T'_4	12.54	(1.73)	11.73	(0.32)	10.78	(4.24)	9.55	(6.66)	13.09	(0.7)	6.48	(-3.4)	6.51	(-2.22)	8.66	(8.66)	6.53	(6.53)	7.76	(-3.29)
	T'_5	13.90	(1.4)	14.09	(1.43)	12.64	(-0.93)	13.42	(-0.41)	7.38	(-1.34)	8.51	(-0.75)	7.86	(-0.04)	1.69	(-8.07)	1.16	(-4.74)	6.29	(0.12)
	T'_6	13.23	(0.44)	13.07	(0.49)	15.44	(0.93)	14.58	(0.35)	13.65	(5.33)	11.29	(3.28)	10.74	(1.85)	11.67	(2.72)	12.15	(4.4)	7.94	(2.43)
	T'_7	13.34	(1.11)	12.97	(1.54)	13.23	(3.99)	15.53	(10.14)	14.12	(4.89)	9.95	(-0.33)	9.81	(-1.88)	4.28	(-3.97)	9.81	(7.19)	2.88	(-7.39)
	T'_8	13.98	(0.17)	13.38	(0.62)	12.68	(-1.21)	13.25	(0.19)	13.46	(0.71)	9.29	(2.28)	9.22	(0.48)	13.47	(10.52)	8.88	(2.76)	5.83	(3.82)
	T'_9	13.45	(0.52)	13.62	(0.39)	6.76	(-3.56)	8.03	(-7.09)	12.30	(-1.19)	7.85	(-0.62)	7.35	(-1.11)	7.83	(-0.49)	7.08	(-2.52)	7.14	(-2.7)
	avg	12.80	/	12.68	/	12.09	/	11.62	/	11.78	/	8.56	/	8.53	/	7.20	/	5.35	/	6.22	/
Zero Shot CoT	T_0	12.44	/	12.62	/	13.61	/	13.39	/	12.25	/	11.83	/	13.55	/	15.90	/	11.89	/	17.30	/
	T_1	12.75	/	13.06	/	13.46	/	13.21	/	14.37	/	13.70	/	13.27	/	11.30	/	15.65	/	19.12	/
	T_2	12.32	/	12.62	/	12.62	/	13.05	/	15.08	/	17.62	/	13.34	/	11.65	/	10.38	/	15.37	/
	T_3	12.04	/	12.05	/	12.82	/	12.27	/	14.28	/	12.54	/	13.09	/	0.00	/	16.26	/	17.07	/
	T_4	12.90	/	13.13	/	13.90	/	13.29	/	14.09	/	14.65	/	10.70	/	15.41	/	10.05	/	16.26	/
	T_5	11.95	/	12.25	/	12.19	/	12.75	/	14.72	/	11.24	/	12.49	/	16.54	/	0.00	/	16.30	/
	T_6	12.04	/	12.15	/	12.49	/	12.53	/	9.85	/	16.03	/	12.60	/	24.13	/	0.00	/	16.95	/
	T_7	12.68	/	12.29	/	13.67	/	12.62	/	12.99	/	14.60	/	16.52	/	11.06	/	9.52	/	17.70	/
	T_8	12.63	/	13.10	/	13.37	/	13.44	/	13.98	/	13.13	/	12.32	/	0.00	/	8.54	/	16.38	/
	T_9	12.70	/	12.46	/	13.13	/	12.74	/	12.79	/	15.24	/	13.41	/	18.75	/	26.70	/	18.43	/
	T'_0	13.08	(0.65)	13.00	(0.39)	13.47	(-0.14)	13.49	(0.11)	11.88	(-0.38)	16.63	(4.8)	14.24	(0.69)	15.89	(-0.02)	16.69	(4.8)	17.11	(-0.19)
	T'_1	12.65	(-0.1)	13.37	(0.31)	13.52	(0.06)	13.01	(-0.2)	11.48	(-2.89)	13.26	(-0.44)	12.82	(-0.45)	15.58	(4.27)	15.42	(-0.23)	17.61	(-1.51)
	T'_2	12.25	(-0.07)	12.71	(0.1)	12.81	(0.19)	12.86	(-0.2)	11.79	(-3.29)	13.09	(-4.53)	14.22	(0.88)	12.68	(1.03)	14.10	(3.72)	16.82	(1.45)
	T'_3	12.58	(0.54)	12.85	(0.8)	13.49	(0.67)	13.41	(1.14)	11.48	(-2.79)	14.23	(1.68)	16.10	(3.01)	22.55	(22.55)	17.89	(1.63)	19.40	(2.33)
	T'_4	13.10	(0.2)	13.32	(0.2)	14.19	(0.29)	14.24	(0.95)	12.55	(-1.54)	15.12	(0.47)	13.41	(2.71)	13.83	(-1.58)	17.23	(7.18)	14.03	(-2.23)
	T'_5	13.03	(1.08)	13.79	(1.54)	13.72	(1.53)	12.83	(0.09)	10.43	(-4.29)	13.58	(2.34)	14.49	(2.0)	14.53	(-2.02)	17.95	(17.95)	18.74	(2.44)
	T'_6	12.82	(0.78)	12.58	(0.43)	13.08	(0.58)	12.80	(0.27)	14.58	(4.72)	13.69	(-2.34)	14.49	(1.9)	12.71	(-11.42)	17.78	(17.78)	16.19	(-0.76)
	T'_7	12.81	(0.13)	12.85	(0.56)	14.97	(1.3)	14.82	(2.2)	0.00	(-12.99)	14.20	(-0.4)	11.92	(-4.59)	12.82	(1.76)	21.34	(11.82)	17.98	(0.28)
	T'_8	12.82	(0.19)	13.08	(-0.02)	13.22	(-0.16)	13.73	(0.29)	15.14	(1.16)	17.82	(4.69)	11.61	(-0.72)	12.49	(12.49)	13.77	(5.23)	14.67	(-1.71)
	T'_9	13.34	(0.64)	13.38	(0.92)	12.87	(-0.26)	12.95	(0.21)	12.57	(-0.22)	13.53	(-1.71)	12.26	(-1.15)	17.36	(-1.39)	20.94	(-5.76)	19.54	(1.11)
	avg	12.65	/	12.83	/	13.33	/	13.17	/	12.32	/	14.29	/	13.34	/	13.76	/	14.10	/	17.15	/
	Overall AVG	12.73	/	12.76	/	12.71	/	12.40	/	12.05	/	11.42	/	10.93	/	10.48	/	9.72	/	11.68	/

Table 7: Fleisch-Kincaid score (`fks`) of the responses generated by the two models and two prompting techniques for each task. The Δ indicates the difference between a task and its corresponding inverted task.

ZS Tasks

Following the previous schema, the results of the Flesch-Kincaid test are presented per quadrant of Table 7. Starting with the upper-left quadrant (GPT-3.5–ZS) of Table 7, the results show significant variability in readability. For example, the scores for the TP group average around 12.8, yet they range from "professional" (10.32 in T_0) to the "college graduate" level (13.98 in T_8'), indicating that some responses are substantially easier to understand than others. This observation is also true for the other groups. For example, the TN group has a notably low minimum value of 6.54 in T_9', indicating an exceptionally complex text (longer words). In contrast, in the FN cases, a higher value of 15.53 can be observed for T_3. Overall, the readability score is usually around the "college graduate" level, with a slight tendency to use more professional English. This is reasonable since these responses are more detailed and contain longer words and sentences, implying lower readability scores. In addition, comparing tasks with their counterparts shows that, for example, the responses of T_0' are significantly easier to understand than T_0 in all groups.

This trend can also be observed in the other direction. For T_9 and T_9' in the FN and TN groups, the readability score changes downward, suggesting that T_9' became more complex to understand (longer words) than T_9 in these categories.

For Llama-2 ZS, the average confusion values are significantly lower (indicating longer words) compared to GPT-3.5 (for example, TP cases: Llama-2 8.56, GPT-3.5 12.32). The same trend is observed for the remaining confusion values and the cases of `NA`. Furthermore, analyzing word lengths, if Llama-2 ZS generates responses exceeding 100 words, they tend to include significantly longer words. When comparing tasks with their counterparts, the average word length remains relatively stable for similar cases, but still reflects longer words overall. However, exceptions are found: T_4 and T_4' show a marked decrease in word length for the TP (-3.4) and FP (-2.22) cases. In contrast, TN and FN scores show increases of 8.66 and 6.53, respectively.

ZS-CoT Tasks

For GPT-3.5 ZS-CoT, the overall averages remain around the same values as for GPT-3.5 ZS, indicating similar word lengths compared to the number of words. For the "positive" values (TP and FP), the averages are 12.65 and 12.83, reflecting a higher readability score compared to ZS. This also applies to the FN and TN values: 13.33 and 13.17, respectively. In general, this suggests that the responses contain moderately more complex words, making them understandable for trained English speakers, as expected given the task's nature. Examining the pair of notable tasks, T_5 and T_5' show a notable decrease in FP (-1.54) and TN (-1.53), suggesting that the responses for T_5' use slightly longer words. However, the drastic difference in T_7 and T_7' (an increase of 12.99) can be attributed to the lack of samples for T_7.

Lastly, the results for Llama-2 ZS-CoT are presented. On average, its readability score is slightly higher than that of other models or methods, with the maximum average in cases of `NA` at 17.15. However, this increase is not sufficient to elevate it into a considerably lower interpretive category. The confusion matrix scores are comparable: TP cases average 14.29, FP cases 13.34, TN cases 13.76, and FN cases 14.11. Analyzing tasks with their counterparts, the FN category shows a notable

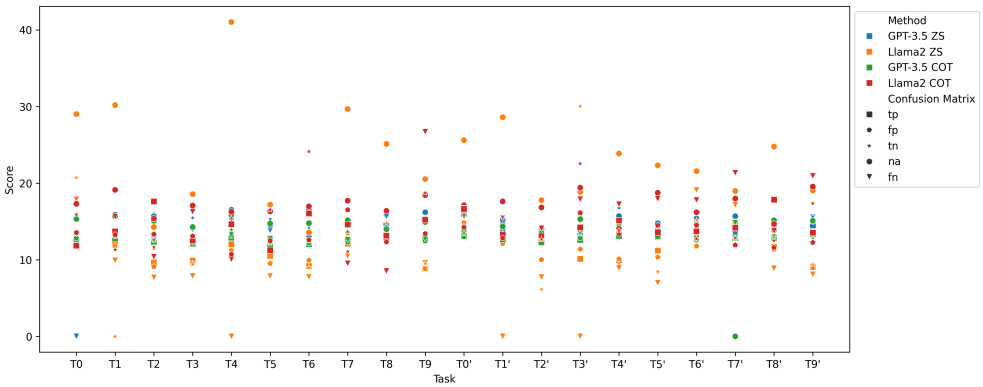

Fig. 7: Visualization of the Fleisch-Kincaid scores shown in Table 2. Different symbols represent the confusion matrix values, whereas different colors indicate the methods and models used.

average decrease (-6.41) in readability compared to other categories, suggesting that the counterparts often became less complex and used longer words (over the total number of words) in this category.

Figure 7 shows a graphical representation of Table 7. A trend can be observed, that for Llama-2 based models the readability scores are unstable. Especially for `NA` cases of Llama-2 ZS the readability scores tend to be higher than all other values.

📌 **Answer to RQ$_2$.** Textual metrics offer valuable insights into LLM responses. However, they fail to link these metrics to actual verdicts, as models respond consistently with the same level of confidence, regardless of the correctness of the verdict.

4 Discussion

This section reflects on the observed results and distills the main take-away messages that could be found by investigating the impact of different task formulations on the performance of LLMs for vulnerability prediction.

4.1 Prompt vs. Actual Verdicts

Generally speaking, one can conclusively state that Large Language Models (LLMs) exhibit a consistent level of confidence in their textual responses, irrespective of the true outcome of the given task. This claim is supported by observed tendencies across different models, regardless of the prompting method used, to favor positive responses predominantly—either as false positives (FP) or true positives (TP). Consequently, it may be beneficial to devise strategies that encourage the model to opt for more frequent expressions of uncertainty, such as "don't know." Such an approach has the

Table 8: Additional tasks that only changed the second part of the CoT prompt.

Task	First Prompt	Second Prompt
$\widetilde{T_1}$	Evaluate the code below for potential vulnerabilities.	Is the code vulnerable?
$\widetilde{T_2}$	Evaluate the code below for potential vulnerabilities.	Can you find a vulnerability?
$\widetilde{T_3}$	Evaluate the code below for potential vulnerabilities.	Tell me if the code is vulnerable.
$\widetilde{T_4}$	Investigate the code below for any possible vulnerabilities.	Is the code vulnerable?
$\widetilde{T_5}$	Investigate the code below for any possible vulnerabilities.	Can you find a vulnerability?
$\widetilde{T_6}$	Investigate the code below for any possible vulnerabilities.	Tell me if the code is vulnerable.

potential to improve prediction accuracy, as it would liberate the model from the binary constraints of a definitive "yes" or "no" reply.

4.2 Impact of Minor Prompt Changes

The study highlighted that minor changes in the prompt text—particularly the part about the task—could lead to noticeable differences in the LLM output despite the semantic meaning remaining the same. To the best of our knowledge, this has never been done in other studies. This underlines the importance of carefully crafting prompts for accurate and consistent predictions. This finding also has a troubling implication. Asking the LLM to solve a task with semantically equivalent prompts but using different formulations could yield very different results. This raises questions about existing studies involving LLMs to predict vulnerabilities and whether the observed results can be trusted. Replicating existing LLM-based studies using alternative but semantically equivalent prompts could assess their validity and reveal potential results that could reframe or even invalidate prior conclusions.

It could be interesting to investigate the sensitivity to the context of the prompt more deeply. This could measure the degree to which LLM performance depends on the exact wording of the input. As a first step in this direction, we also briefly investigated the impact of rephrasing the second part of the prompt used in the ZS-CoT technique as a post hoc analysis. In our case, we rephrased the (`<FOLLOW-UP>`) task (as described in Section 2.4) for the best and worst tasks that GPT-3.5 had with ZS-CoT, that is, we did not consider Llama-2 due to its bad results with ZS-CoT. The new tasks are reported in Table 8. However, this analysis was performed for both models. The results of this sub-study are shown in Table 9. The results indicate a significant variation in the F1 score for both models. Ranging from 0.56 to 0.65 for GPT-3.5 and 0.61 to 0.69 for Llama-2. However, the previous observations still hold, since Llama-2 produces nearly 10 times as many `NA` cases as GPT-3.5. Therefore, the significance of these results is limited. Having such a significant range with small changes in the prompt also underscores the need to replicate existing LLM-based vulnerability prediction studies.

Additionally, we observed substantial differences in the TP and FP values of certain pairs of tasks and their inverted versions, such as between T_4 vs. T_4' and T_6 vs. T_6' within GPT-3.5 ZS-CoT. To shed light on these discrepancies, we conducted a manual analysis of the input samples and responses for T_4 vs. T_4' and T_6 vs. T_6', particularly where the verdicts changed (i.e., the classification for T_i was "non-vulnerable", while for T_i' was "vulnerable"). For the pair T_4–T_4', we found that 49 cases had differing verdicts,

Table 9: Performance of the additional CoT tasks.

	GPT-3.5								Llama-2							
Task	TP	FP	TN	FN	NA	Pr	Re	F1	TP	FP	TN	FN	NA	Pr	Re	F1
$\widetilde{T_1}$	50	48	24	20	12	0.51	0.71	0.60	41	44	5	6	58	0.48	0.87	0.62
$\widetilde{T_2}$	72	72	3	3	4	0.50	0.96	0.66	26	29	0	4	95	0.47	0.87	0.61
$\widetilde{T_3}$	51	47	23	24	9	0.52	0.68	0.59	57	48	4	4	41	0.54	0.93	0.69
$\widetilde{T_4}$	48	47	24	27	8	0.51	0.64	0.56	45	42	8	6	53	0.52	0.88	0.65
$\widetilde{T_5}$	68	66	9	9	2	0.51	0.88	0.64	27	27	0	2	98	0.50	0.93	0.65
$\widetilde{T_6}$	46	45	25	27	11	0.51	0.63	0.56	47	47	6	8	46	0.50	0.85	0.63
Total	335	325	108	110	46	0.51	0.75	0.61	243	237	23	30	391	0.51	0.89	0.65

Table 10: Best 3 performing ZS and ZS-CoT tasks applied to state-of-the-art LLMs.

		Qwen2.5-Coder								DeepSeek-Coder							
	Task	TP	FP	TN	FN	NA	Pr	Re	F1	TP	FP	TN	FN	NA	Pr	Re	F1
ZS	T_0	67	68	6	7	6	0.50	0.91	0.64	66	55	5	1	27	0.55	0.99	0.70
	T_2	72	74	2	4	2	0.49	0.95	0.65	61	64	4	4	21	0.49	0.94	0.64
	T_7'	61	62	6	12	13	0.50	0.84	0.62	48	46	11	8	41	0.51	0.86	0.64
	Average	66.66	68.00	4.66	7.66	7.00	0.50	0.90	0.64	58.33	55	6.66	4.33	29.66	0.51	0.93	0.66
ZS-CoT	T_3	76	68	8	1	1	0.53	0.99	0.69	72	74	2	5	1	0.49	0.94	0.65
	T_0'	73	75	1	2	3	0.49	0.97	0.65	61	57	19	15	2	0.52	0.80	0.63
	T_4'	74	73	0	0	7	0.50	1.00	0.67	53	46	16	16	23	0.54	0.77	0.63
	Average	74.33	72.00	3.00	1.00	3.66	0.51	0.99	0.67	62.00	59.00	12.33	12.00	8.66	0.51	0.84	0.64

with 85% (42 cases) exhibiting opposing results, i.e., a task gave "vulnerable" verdict while the other gave "non-vulnerable" verdict. In the remaining seven instances, one of the two tasks in the pair did not yield a clear verdict (i.e., "NA"). A similar pattern emerged for the pair T_6–T_6', with 82% of the 63 cases showing divergent results, including 11 pairs with an "NA" verdict. This analysis highlights that minor task alterations, such as inversion, can significantly affect detection performance and may even yield totally different results.

4.3 Using State-of-the-Art LLMs

To bridge the gap between the evaluation of our foundational approaches and modern LLMs, we adapt and validate the top 3 tasks for both ZS and ZS-CoT prompting on leading large language models, based on Hugging Face's Big Code Models Leaderboard for Java,[10] which evaluates (among other) the code understanding properties of LLMs. From this leaderboard, we selected the two main foundational models: Qwen2.5-Coder and DeepSeek-Coder.

The top half of Table 10 presents the results for the ZS prompting applied to Qwen2.5-Coder and DeepSeek-Coder. In this case, the best tasks were T_0, T_2, and T_7'. Qwen2.5-Coder achieved an average recall of 0.90, with results ranging from 0.84 in T_7' to 0.95 in T_2. The precision of Qwen2.5-Coder is moderate at an average of 0.5, while its overall F1 score is 0.64, remaining rather consistent for all tasks. In the end, the results for Qwen2.5-Coder are not too different than those seen for "older" models

[10] https://huggingface.co/spaces/bigcode/bigcode-models-leaderboard

Llama-2 and GPT-3.5. The results of DeepSeek-Coder are quite similar. Its precision has an average of 0.51, with the average recall slightly (but negligibly) better, reaching 0.93. This resulted in an average F1 score of 0.66, reaching 0.70 for T_0. In contrast to that, DeepSeek-Coder produces a significantly higher number of "NA" responses than Qwen2.5-Coder ($\sim$ 4 times more on average), indicating that it might be harder to get a clean prediction from DeepSeek-Coder, but overall better results, when a clear response was given.

The bottom half of Table 10 shows the results for ZS-CoT applied to Qwen2.5-Coder and DeepSeek-Coder. In this case, the best tasks were T_3, T_0', and T_4'. When analyzing the performance of the given tasks, a clear pattern emerges: Qwen2.5-Coder captures almost every positive instance, producing exceptionally high recall rates of 0.97 on T_0', and even perfect on T_4', resulting in an average of 0.99. As a result, it fails to miss virtually any true positives (only three total false negatives out of three tasks), but this comes at the cost of nearly indiscriminate positive prediction. The precision, however, stands on 0.5 on average, not much different than the ZS case. These shortcomings in precision lead, in the end, to a comparable F1 score to the older models Llama-2 and GPT-3.5. DeepSeek-Coder, in contrast, was much better in classifying true negatives (37 in total across the three tasks), but its recall dropped to an average of 0.84, i.e., 0.15 less than Qwen2.5-Coder. DeepSeek-Coder too stopped at an average precision of 0.51; hence, its average F1 score was lower than Qwen2.5-Coder due to its lower average recall.

In practical terms, one should prefer Qwen2.5-Coder if missing a true positive carries a high cost and you can tolerate frequent false alarms; DeepSeek-Coder is a better fit when a more reliable negative judgment, fewer false positives are required. Overall, they are not much different from smaller models like Llama-2 and GPT-3.5, which we assessed in the main part of the study.

4.4 Characteristics of Responses

There are no clear or directly noticeable patterns in response quality related to correctness, besides positive classifications (TP and FP) produce longer responses (for example, TP and FP responses are $\sim$500 characters longer than TN, FN, and NA for GPT-3.5–ZS). However, our analysis only scratched the surface of possible analysis methods. Analyzing these properties further can provide insight into improving LLM performance. For example, leveraging the content of each response with "thematic analysis" or "topic modeling" could be used to find better correlations between specific words used and the actual analysis verdict. These methodologies are designed to discern and elucidate persistent patterns or motifs within responses. In doing so, they may also facilitate the discovery of deeper and more comprehensive insights.

4.5 Analysis of Failure Patterns

To underline the characteristics of the responses, we also searched for potential patterns in the cases where the models had a "NA" verdict. Namely, we manually inspected and categorized the "NA" responses of the best three tasks for the best-performing model GPT-3.5 for both ZS and ZS-CoT—the same also involved in

Table 11: Number of times a response ended up with the four pattern of "NA" verdict responses.

	Task	CD	MC	MI	CV
ZS	T_0	4	4	0	0
	T_2	0	6	0	1
	T_7'	0	7	0	3
ZS-CoT	T_0'	0	4	0	0
	T_3	0	3	1	0
	T_4'	0	7	0	0

Section 4.3. The results are presented in Table 11. The categories were created using an inductive approach: The categories were created while reviewing the responses. This leads to the following four categories:

- **CD** – ***Clear but Doubt***. The response contains a clear verdict, but it also expresses doubts on its own verdict, e.g., *'there are no obvious vulnerabilities, it is difficult to definitively determine if there are any vulnerabilities'*.
- **MC** – ***Missing Context***. The response contains a phrase indicating that more information/context is needed to asses the vulnerability, e.g. *'without the full context of the application and its usage, it's difficult to definitively determine if there are any vulnerabilities'*.
- **MI** – ***Missing Implementation***. The response calls for the exact implementation of some elements of the method analyzed, e.g., *'it's difficult to definitively identify the number of vulnerabilities without a comprehensive understanding of the entire system and the implementation details of the KeyStoreHelperUtil'*.
- **CV** – ***Code explanation without Verdict***. The response only summarizes or describes the lines of the given method without assessing potential vulnerabilities, and remains rather vague.

As presented in Table 11, the vast majority of "NA" cases for both prompting types fell into the MC (Missing Context) category, indicating the need for additional context to assess the vulnerability fully. This was rather expected since the analyzed method often invoked various other methods, which were not part of the given context. Therefore, it remains very challenging for the model to understand the full context if the full source code is not recognized. The MI (Missing Implementation) case for T_3 in ZS-CoT can be viewed as a specialized case of MC, since the response calls for further detail, but on a specific part of the code. The CD (Clear but Doubt) follows a similar pattern, since the responses indicate the absence of obvious vulnerabilities; however, it cannot guarantee it due to missing information about the rest of the system/program. This happened four times for task T_0 in the ZS case. For CV (Code explanation without Verdict) cases, the response contains a very general vulnerability assessment that is not specific to the given code. In addition, the parts that are specific only summarize or describe the lines presented in the input. This happened three times for task T_7' in the ZS case.

In summary, our inductive analysis of the "NA" responses revealed that most of the model's uncertainty stems from insufficient context: Without access to the complete codebase or implementation details, GPT-3.5 frequently defers judgment. A smaller subset of cases reflects more specific implementation gaps (MI) or cautious hedging despite apparent clarity (CD). At the same time, purely descriptive answers (CV) underscore its tendency to explain rather than evaluate when uncertain. These findings highlight the critical role of visibility in the full context of reliable vulnerability assessment.

4.6 Vulnerability Severity Analysis

To further discuss the impact on real-world scenarios, we perform a vulnerability context analysis. Therefore, we mapped all samples from the input dataset to their respective CVSS v4.0 scoring.[11] This rating system categorizes all CVEs into four different severity levels: LOW, MEDIUM, HIGH, and CRITICAL. Next, we analyze the detection performance in each severity category for the 4 models in the top 3 tasks per promoting technique—the same also used in Section 4.3. The results of this analysis can be found in Table 12. The findings demonstrate that across all models examined, particularly within the severity categories, the number of True Positive (TP) and False Positive (FP) instances is approximately equivalent. This suggests that there is effectively a 50:50 probability of correctly identifying high-risk vulnerabilities when the model claims to have found one, independent of their severity. This detection rate is further supported by the precision metrics of the overall detection performance (from Table 3). Furthermore, it is important to note that for both Llama-2 (ZS-CoT) and DeepSeek-Coder (ZS), the significant number of cases `NA` complicates the ability to draw definitive conclusions from this analysis. In particular, T_3 on Llama-2 appears to encounter the most significant challenges in assessing high-severity vulnerabilities, with 48 cases marked as `NA`. In contrast, DeepSeek-Coder faces the greatest difficulty with high-severity vulnerabilities in T_2', where 23 cases were designated as `NA`.

4.7 Challenges in Automated Evaluation

A significant limitation in the use of conversational LLMs such as GPT-3.5 and Llama-2 for vulnerability prediction is the challenge of establishing a simple, fully automated response evaluation process. Despite deterministic configurations, such as setting the temperature parameter to zero, variations in output can still occur due to inherent model uncertainties and contextual nuances of prompts. These inconsistencies require manual inspection to ensure accurate interpretation and labeling, as observed in this study, which involved extensive human intervention. Evaluating responses is difficult due to the variability and occasional inconclusive output of the models, requiring manual oversight to validate and interpret responses.

To address this, we propose a "response radar" tool concept to extract the gist of the responses. This tool concept would filter out irrelevant or verbose content and focus on providing concise summaries to assist reviewers in making final decisions.

[11] https://nvd.nist.gov/vuln-metrics/cvss

			GPT-3.5					Llama-2					Qwen2.5-Coder					DeepSeek-Coder				
	Severity	Task	TP	FP	TN	FN	NA	TP	FP	TN	FN	NA	TP	FP	TN	FN	NA	TP	FP	TN	FN	NA
ZS	Critical	T0	13	12	0	0	3	13	13	1	1	0	13	14	0	0	1	14	12	0	0	2
		T2	11	10	3	3	1	9	9	0	1	9	14	13	0	0	1	12	12	0	0	4
		T7'	13	13	0	0	2	9	9	1	1	8	14	13	0	0	1	12	11	2	0	3
	High	T0	29	25	5	0	1	20	21	5	4	10	26	27	2	3	2	27	23	1	0	9
		T2	27	26	1	3	3	23	23	2	2	10	27	29	1	2	1	27	26	1	0	6
		T7'	27	22	5	2	4	23	20	4	3	10	20	24	1	7	8	14	14	4	5	23
	Medium	T0	28	26	1	0	3	19	20	3	4	12	26	23	4	2	3	23	18	3	1	13
		T2	27	23	4	1	3	18	19	3	1	17	27	28	1	2	0	21	24	2	3	8
		T7'	24	21	5	4	4	19	22	1	3	13	24	22	4	4	4	20	18	4	3	13
	Low	T0	1	2	1	1	1	3	3	0	0	0	1	3	0	2	0	2	2	1	0	1
		T2	0	2	1	3	0	3	2	1	0	0	3	3	0	0	0	0	1	1	1	3
		T7'	1	3	0	2	0	3	3	0	0	0	2	2	1	1	0	1	2	1	0	2
ZS-CoT	critical	T3	12	12	1	2	1	3	3	0	2	20	13	11	2	1	1	13	13	0	1	1
		T0'	11	8	4	3	2	6	6	1	1	14	13	13	0	0	2	10	13	0	4	1
		T4'	12	11	2	2	1	4	5	0	1	18	13	11	0	0	4	10	10	2	3	3
	high	T3	27	22	7	3	1	6	6	0	0	48	30	28	2	0	0	28	28	2	2	0
		T0'	22	19	11	6	2	14	21	2	4	19	28	30	0	1	1	23	20	10	7	0
		T4'	27	28	2	3	0	7	9	0	1	43	28	29	0	0	3	22	18	8	4	8
	medium	T3	24	23	5	4	2	9	6	0	0	43	29	26	3	0	0	27	29	0	2	0
		T0'	15	18	8	9	8	18	16	2	1	21	28	28	1	1	0	25	20	9	3	1
		T4'	27	24	4	2	1	14	12	0	0	32	29	29	0	0	0	18	15	5	8	12
	low	T3	2	1	2	1	0	2	0	0	0	4	3	3	0	0	0	3	3	0	0	0
		T0'	1	2	1	2	0	3	1	0	0	2	3	3	0	0	0	2	3	0	1	0
		T4'	1	2	1	2	0	2	1	0	0	3	3	3	0	0	0	2	2	1	1	0

Table 12: Vulnerability severity analysis showing the detection performance by severity category (CVSS v4.0 Ratings) of the top 3 tasks (based on performance) for all 4 LLMs.

Given the dynamic and verbose nature of conversational LLM outputs, distinguishing meaningful insights from extraneous or low-value content is a significant challenge. We observed that responses often included verbose reasoning, irrelevant tangents, or statements of uncertainty, which can obscure the core message. This necessitates the development of a mechanism designed to analyze the gist of the model's response, filtering out noise and irrelevant elements to improve the interpretability of output. This tool concept acts as a sort of "*response radar*" that would serve as an intermediary analytical layer, processing model-generated responses to identify and highlight their essential components. This system could leverage Natural Language Processing (NLP) models, such as fine-tuned LLMs or other state-of-the-art natural language understanding methods. The tasks of this *response radar* could be:

- **Gist Extraction**: Summarize the primary message or conclusion of the response, focusing on key phrases or direct answers relevant to the task (e.g. "The code is vulnerable due to reason X").
- **Filter Noise**: Identify and prioritize content flagged as extraneous or irrelevant, such as verbose reasoning, redundant explanations, or overuse of hedging language ("might," "possibly," and similar terms).
- **Quantify Confidence**: Provide metrics or indicators of the response's confidence and clarity by assessing patterns in the language, such as the use of definitive statements versus ambiguous qualifiers.
- **Highlight Actionable Insights**: Emphasize actionable suggestions or findings, such as specific vulnerabilities or improvements mentioned, while suppressing overly generic advice or conceptual filler.

Developing an effective *response radar* requires tuning its model on a corpus of annotated responses, distinguishing high-value content from noise as a starting point, the manually reviewed responses of our study can be used to this end. In addition, integrating feedback loops from human reviewers could refine its accuracy over time. A well-designed system could significantly enhance the usability and reliability of conversational LLMs in complex tasks like vulnerability prediction by acting as a filter and guide to interpret LLM responses.

Furthermore, models often produce responses with "extraneous" elements, such as irrelevant information, redundant explanations, and characters in lines new (\n). These artifacts complicate the automated parsing and assessment of the outputs. For example, while models attempt to identify vulnerabilities, they frequently hedge their assertions using uncertain terms like "might" or "potentially," which require nuanced interpretation to infer the intended classification (e.g., "vulnerable" or "not vulnerable"). The variability and uncertainty in model reasoning also highlight limitations in their robustness and reliability, particularly when the prompt structure changes slightly. This lack of consistency in reasoning, coupled with lengthy or tangential content, underscores the need for iterative prompt refinement and manual validation of outputs. These challenges prevent the adoption of a fully automated pipeline for vulnerability prediction and require a hybrid approach that combines automated initial evaluations with human oversight for the final judgment.

4.8 Guidelines to Improve Task Performance

The results of this study could also be useful for engineers and security specialists. Therefore, a series of heuristics can be derived to support prompt engineering in vulnerability detection. First, tasks should be concise and straightforward, as too detailed instructions tend to lower readability without enhancing classification performance. In addition, using ZS-CoT can improve reasoning by encouraging analytical thinking, but should be accompanied by clear follow-up instructions to minimize ambiguous or incomplete responses. On another note, inverted tasks generally perform lower, particularly with GPT-3.5, and should be avoided unless thoroughly validated in controlled environments. Also, querying the same prompt multiple times and employing majority voting among the outputs increased stability and minimized random variations, offering a straightforward yet effective strategy for deployment. Practitioners should also be cautious of too brief or nonspecific tasks, which frequently resulted in inconsistent or poor quality responses. Together, these strategies present a set of practical practices that equip engineers to design more reliable triggers and effectively use LLMs for vulnerability assessment.

5 Threats to Validity

Sampling biases: The manual evaluation process may be susceptible to sampling bias if the selection of vulnerability instances is not representative of the general population of vulnerabilities. Bias may arise if certain types of vulnerability are overrepresented or underrepresented in the evaluation dataset, leading to skewed conclusions about the model's performance. Using a single batch of 154 Java methods may not provide a comprehensive view of all possible vulnerabilities. The variability in the types of vulnerability across the data sets could impact the effectiveness of the model when generalized to other software contexts.

Inter-Rater Reliability: The reliability of manual evaluation depends on the agreement among different evaluators to identify and label vulnerabilities. Lack of consistency between evaluators in recognizing and categorizing vulnerabilities can undermine the reliability of the evaluation results, introducing uncertainty into the evaluation of model performance.

Temporal Bias: The evaluation data set may not reflect the temporal evolution of vulnerabilities over time. The vulnerabilities present in the evaluation dataset may differ in prevalence, severity, or characteristics from those encountered in current or future contexts, potentially biasing the evaluation results and limiting the applicability of the findings to contemporary settings.

Internal Validity: Prompt Interpretation and Labeling: Since responses were manually labeled according to interpretation, subjective biases could influence label assignment, particularly in ambiguous cases. The manual interpretation process, even with consistency checks, might introduce errors or inconsistencies.

Repetition Consistency: Although the prompts were repeated to ensure consistency, the variability in some model responses (particularly ChatGPT-3.5) may affect internal consistency, which could limit the reproducibility of certain results.

Construct Validity: Prompt Rephrasing as a Performance Measure: The study assumes that prompt rephrasing significantly impacts model performance in vulnerability detection. However, other factors, such as model configuration or inherent biases in LLMs, could also play a role. This reliance on prompt formulation alone may not capture all the variables that impact accuracy. Furthermore, we also analyzed other metrics like cosine similarity, word and punctuation count, and various other readability scores (found in the (online) appendix). However, these analyzes did not provide deeper insight or uncover new patterns that were not already discovered by the metrics presented in this article.

External Validity: The results may not be generalized to other LLMs, programming languages, or types of software vulnerabilities.

The selection of LLMs may influence the outcomes of this study, complicating the ability to generalize the findings about the observed performance to all kinds of large language models. Further analysis of two state-of-the-art models, i.e., Qwen2.5-Coder and DeepSeek-Coder, indicated that Llama-2 and GPT-3.5 operate within a comparable range, allowing the same conclusions regarding trends and the influence of minor prompt alterations to also hold for other more advanced models.

6 Related Work

This section provides a review of prior studies, methodologies, and frameworks relevant to our research. This section compares our work within the existing body of knowledge, by examining recent advancements and key studies of two general categories: First, LLM- and prompt-based vulnerability prediction and second, research focusing on the impact of rephrasing in prompt based LLMs.

Cheshkov et al. [22] evaluates the performance of GPT-3 models for code vulnerability prediction in Java. They analyzed the performance capabilities for binary and multi-label CWE classification. For the binary classification they used prompts like "write whether the vulnerability is contained in the code in the Yes/no format" or "write whether this function is vulnerable:". For the multi-class labeling prompts like: "Which of these types of vulnerabilities is contained in the code?" plus a list of possible CWE classes. In their experiments GPT-3.5-turbo had a F1-score of 0.62 which is lower then the predecessor text-davinci-003 with 0.67 f1-score. The authors also point out that CoT prompting could also be an interesting direction to continue.

Jensen et al. [23] also created an evaluation of LLMs for vulnerability prediction. Hereby, they used a large variety of models (9 in total) on various datasets. To construct the used prompts, they utilized zero-shot and chain-of-though prompts. When applied to flagging security vulnerabilities, the top proprietary model achieves an accuracy exceeding 95.6% and an F1 score above 37.9%. For software functionality validation, the model reaches an accuracy and F1 score over 88.2%. Additionally, the model's vulnerability descriptions align with actual vulnerabilities more than 36.7% of the time.

Yin et al. [24] tested state-of-the-art vulnerability detection (11 in total) LLMs and additionally, fine-tuned them for vulnerability detection, assessment, location and description tasks. The results are ranging from under 10% in f1-score up to the

best performing model (SVulD [25]) having a f1-score of 33.6% for the vulnerability detection task.

Although, researchers use LLMs in various vulnerability applications including vulnerability prediction, the related work lacks to provide a systematic source for the prompts or task definitions used in these applications, especially a detailed discussion of the impact of different phrasing. This shows a gap in research leading to this work.

7 Conclusion

In this study, we explore how prompt phrasing affects the accuracy and reliability of large language models (LLMs) in the context of vulnerability prediction. Further, how the LLM's response behavior changes based on differently phrased semantically equal tasks. By systematically varying the task formulations and employing advanced techniques such as Zero-Shot and Zero-Shot Chain-of-Thought (ZS CoT), we evaluated the performance of two prominent LLMs (ChatGPT-3.5, LLaMA2) on their ability to detect vulnerabilities in Java code. Our findings reveal that subtle differences in task phrasing can substantially influence model predictions, indicating a sensitivity in LLMs to linguistic properties that extend beyond traditional prompt engineering insights.

In particular, our results indicate that prompt rephrasing can significantly impact classification outcomes, with particular formulations that produce improved prediction metrics such as precision, recall, and F1 score. This leads to inquiries regarding current research using LLMs for vulnerability prediction, and whether the outcomes observed are reliable and reproducible. However, this needs to be investigated further.

Furthermore, we present a novel approach to correlating textual metics with the outcome of prediction verdicts. However, we were unable to identify a clear pattern besides a connection between positive predictions (TP and FP) and the length of the response. The analysis showed the necessity of further investigations into more profound textual analysis like "thematic analysis" or "topic modeling". In addition, we proposed additional studies that involve the development of a "response radar" that can help LLM response reviewers to highlight important parts of a response or filter out noise.

This research underscores the importance of prompt phrasing as a factor in maximizing the efficacy of LLMs for security-related tasks. By refining prompt strategies and recognizing the effects of linguistic variation, developers and researchers can better leverage LLMs for practical applications in vulnerability detection. Future work could further refine these findings by building on these insights and creating a predictor based on the linguistic properties of responses, ultimately leading to the development of more robust and reliable LLM-based tools for software vulnerability prediction. Besides, we plan to integrate the findings of this analysis by widening the scope to other programming languages like C/C++ or Python and involve a wider set of vulnerabilities for Java to confirm the patterns observed here. In particular, we plan to examine the differences and peculiarities across the various languages to observe whether certain prompt styles work better in specific circumstances. To enable this extension without demanding a high human effort, we will design an automated assessment mechanism

of the LLM responses to relieve the load of the inspectors in determining the models' exact predictions.

Data Availability

Data supporting this study are openly available at https://anonymous.4open.science/r/Beyond-Prompting-BA15/.

Acknowledgements

This work was partially supported by EU-funded project Sec4AI4Sec (grant no. 101120393).

References

[1] GitHub: Top 50 Programming Languages Globally . https://innovationgraph.github.com/global-metrics/programming-languages#programming-languages-rankings. Online; accessed 20 May 2025 (2025)

[2] Veracode: Annual Report on the State of Software Security. https://www.veracode.com/state-software-security-2024-report/. Online; accessed 20 May 2025 (2024)

[3] Ponta, S.E., Plate, H., Sabetta, A., Bezzi, M., Dangremont, C.: A Manually-Curated Dataset of Fixes to Vulnerabilities of Open-Source Software. arXiv. arXiv:1902.02595 [cs] (2019). http://arxiv.org/abs/1902.02595 Accessed 2024-02-20

[4] Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., Neubig, G.: Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. ACM Comput. Surv. **55**(9) (2023) https://doi.org/10.1145/3560815

[5] Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., Zhou, D.: Self-Consistency Improves Chain of Thought Reasoning in Language Models (2023). https://arxiv.org/abs/2203.11171

[6] Ye, X., Durrett, G.: The unreliability of explanations in few-shot prompting for textual reasoning. In: Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., Oh, A. (eds.) Advances in Neural Information Processing Systems, vol. 35, pp. 30378–30392. Curran Associates, Inc., ??? (2022). https://proceedings.neurips.cc/paper_files/paper/2022/file/c402501846f9fe03e2cac015b3f0e6b1-Paper-Conference.pdf

[7] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., Zhou, D.: Chain-of-Thought Prompting Elicits Reasoning in Large Language Models (2023)

[8] Kojima, T., Gu, S.S., Reid, M., Matsuo, Y., Iwasawa, Y.: Large Language Models are Zero-Shot Reasoners (2023)

[9] Steenhoek, B., Rahman, M.M., Roy, M.K., Alam, M.S., Barr, E.T., Le, W.: A Comprehensive Study of the Capabilities of Large Language Models for Vulnerability Detection (2024). https://arxiv.org/abs/2403.17218

[10] Boonthum, C.: iSTART: Paraphrase recognition. In: Proceedings of the ACL Student Research Workshop, pp. 31–36. Association for Computational Linguistics, Barcelona, Spain (2004). https://aclanthology.org/P04-2006

[11] Dong, Q., Wan, X., Cao, Y.: ParaSCI: A large scientific paraphrase dataset for longer paraphrase generation. In: Merlo, P., Tiedemann, J., Tsarfaty, R. (eds.) Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, pp. 424–434. Association for Computational Linguistics, Online (2021). https://doi.org/10.18653/v1/2021.eacl-main.33 . https://aclanthology.org/2021.eacl-main.33

[12] Mallinson, J., Sennrich, R., Lapata, M.: Paraphrasing revisited with neural machine translation. In: Lapata, M., Blunsom, P., Koller, A. (eds.) Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers, pp. 881–893. Association for Computational Linguistics, Valencia, Spain (2017). https://aclanthology.org/E17-1083

[13] Grammarly: How to Paraphrase (Without Plagiarizing a Thing). website. https://www.grammarly.com/blog/summarizing-paraphrasing/paraphrase/ (2024)

[14] Narayan, S., Gardent, C., Cohen, S.B., Shimorina, A.: Split and rephrase. In: Palmer, M., Hwa, R., Riedel, S. (eds.) Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pp. 606–616. Association for Computational Linguistics, Copenhagen, Denmark (2017). https://doi.org/10.18653/v1/D17-1064 . https://aclanthology.org/D17-1064

[15] Aladics, T., Hegedűs, P., Ferenc, R.: A comparative study of commit representations for jit vulnerability prediction. Computers **13**(1) (2024) https://doi.org/10.3390/computers13010022

[16] Zhang, C., Liu, H., Zeng, J., Yang, K., Li, Y., Li, H.: Prompt-enhanced software vulnerability detection using chatgpt. In: Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings. ICSE-Companion '24, pp. 276–277. Association for Computing Machinery, New York, NY, USA (2024). https://doi.org/10.1145/3639478.3643065 . https://doi.org/10.1145/3639478.3643065

[17] Cohen, J.: A coefficient of agreement for nominal scales. Educational and Psychological Measurement **20**(1), 37–46 (1960) https://doi.org/10.1177/

001316446002000104

[18] McHugh, M.: Interrater reliability: The kappa statistic. Biochemia medica : časopis Hrvatskoga društva medicinskih biokemičara / HDMB **22**(3), 276–82 (2012) https://doi.org/10.11613/bm.2012.031

[19] Powers, D.M.W.: Evaluation: From precision, recall and f-measure to roc., informedness, markedness & correlation. Journal of Machine Learning Technologies **2**(1), 37–63 (2011)

[20] Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. ACM Press Books. ACM Press, ??? (1999)

[21] Flesch, R.: A new readability yardstick. J Appl Psychol **32**(3), 221–233 (1948)

[22] Cheshkov, A., Zadorozhny, P., Levichev, R.: Evaluation of chatgpt model for vulnerability detection. arXiv preprint arXiv:2304.07232 (2023)

[23] Jensen, R.I.T., Tawosi, V., Alamir, S.: Software vulnerability and functionality assessment using llms. In: 2024 IEEE/ACM International Workshop on Natural Language-Based Software Engineering (NLBSE), pp. 25–28 (2024). IEEE

[24] Yin, X., Ni, C., Wang, S.: Multitask-based evaluation of open-source llm on software vulnerability. IEEE Transactions on Software Engineering, 1–16 (2024) https://doi.org/10.1109/TSE.2024.3470333

[25] Ni, C., Yin, X., Yang, K., Zhao, D., Xing, Z., Xia, X.: Distinguishing Look-Alike Innocent and Vulnerable Code by Subtle Semantic Representation Learning and Explanation (2023). https://arxiv.org/abs/2308.11237

Appendix A Additional Response Analysis

As mentioned in Section 3.2, we performed additional textual analysis of the responses. Across the tables, we assess the two models under two prompting strategies on various tasks: Table 1 compares ten-run majority-voting performance to single-run scores (with ΔS quantifying that gap), Table 2 reports the average cosine similarity among the ten outputs (with Δ indicating the difference between each task and its inversion), Table 3 tabulates punctuation counts (again using Δ for the task–inversion contrast), Table 4 gives Coleman–Liau readability indices (with Δ between original and inverted tasks), and Table 5 presents Automated Readability Index (ARI) values (with Δ denoting the same original–inversion differences).

		GPT-3.5															Llama-2													
	Task	TP	ΔS	FP	ΔS	TN	ΔS	FN	ΔS	NA	ΔS	Pr	ΔS	Re	ΔS	F1	ΔS	TP	ΔS	FP	ΔS	TN	ΔS	FN	ΔS	NA	ΔS	Pr	ΔS	Re	ΔS	F1	ΔS
Zero Shot	T_0	72.00	-1.00	66.00	3.00	7.00	-3.00	1.00	1.00	8.00	0.00	0.52	-0.01	0.99	-0.01	0.68	-0.02	55.00	0.00	57.00	0.00	9.00	0.00	9.00	0.00	24.00	0.00	0.49	0.00	0.86	0.00	0.62	0.00
	T_1	55.00	-2.00	52.00	0.00	15.00	0.00	16.00	1.00	16.00	1.00	0.51	-0.01	0.77	-0.02	0.62	-0.01	54.00	0.00	56.00	0.00	8.00	0.00	10.00	0.00	26.00	0.00	0.49	0.00	0.84	0.00	0.62	0.00
	T_2	66.00	-2.00	62.00	3.00	9.00	-1.00	10.00	0.00	7.00	0.00	0.52	-0.02	0.87	-0.00	0.65	-0.02	53.00	0.00	53.00	0.00	6.00	0.00	4.00	0.00	38.00	0.00	0.50	0.00	0.93	0.00	0.65	0.00
	T_3	67.00	3.00	68.00	0.00	4.00	2.00	9.00	-2.00	6.00	-3.00	0.50	0.01	0.88	0.03	0.64	0.02	57.00	0.00	52.00	0.00	6.00	0.00	3.00	0.00	36.00	0.00	0.52	0.00	0.95	0.00	0.67	0.00
	T_4	45.00	-5.00	40.00	-2.00	31.00	2.00	26.00	6.00	12.00	-1.00	0.53	-0.02	0.63	-0.08	0.58	-0.04	50.00	0.00	50.00	0.00	14.00	0.00	14.00	0.00	26.00	0.00	0.50	0.00	0.78	0.00	0.61	0.00
	T_5	66.00	-2.00	64.00	1.00	9.00	-1.00	10.00	1.00	5.00	1.00	0.51	-0.01	0.87	-0.02	0.64	-0.01	51.00	0.00	53.00	0.00	7.00	0.00	4.00	0.00	39.00	0.00	0.49	0.00	0.93	0.00	0.64	0.00
	T_6	68.00	1.00	67.00	1.00	8.00	-2.00	9.00	-1.00	2.00	1.00	0.50	-0.00	0.88	0.01	0.64	0.00	55.00	0.00	57.00	0.00	3.00	0.00	2.00	0.00	37.00	0.00	0.49	0.00	0.96	0.00	0.65	0.00
	T_7	66.00	-2.00	64.00	2.00	8.00	-1.00	8.00	2.00	8.00	-1.00	0.51	-0.02	0.89	-0.03	0.65	-0.02	60.00	0.00	60.00	0.00	4.00	0.00	4.00	0.00	26.00	0.00	0.50	0.00	0.94	0.00	0.65	0.00
	T_8	60.00	1.00	58.00	-3.00	14.00	4.00	12.00	0.00	10.00	-2.00	0.51	0.02	0.83	0.00	0.63	0.01	61.00	0.00	59.00	0.00	5.00	0.00	4.00	0.00	25.00	0.00	0.51	0.00	0.94	0.00	0.66	0.00
	T_9	64.00	0.00	64.00	0.00	9.00	1.00	11.00	-1.00	6.00	0.00	0.50	0.00	0.85	0.01	0.63	0.00	49.00	0.00	45.00	0.00	8.00	0.00	4.00	0.00	48.00	0.00	0.52	0.00	0.92	0.00	0.67	0.00
	T'_0	61.00	1.00	57.00	3.00	12.00	-2.00	13.00	-2.00	13.00	-2.00	0.52	-0.01	0.85	0.00	0.64	-0.01	62.00	0.00	59.00	0.00	2.00	0.00	0.00	0.00	31.00	0.00	0.51	0.00	1.00	0.00	0.68	0.00
	T'_1	53.00	0.00	45.00	2.00	25.00	-1.00	17.00	1.00	14.00	-2.00	0.54	-0.01	0.76	-0.01	0.63	-0.01	61.00	0.00	60.00	0.00	2.00	0.00	1.00	0.00	30.00	0.00	0.50	0.00	0.98	0.00	0.67	0.00
	T'_2	52.00	2.00	53.00	1.00	20.00	-2.00	22.00	-2.00	7.00	1.00	0.50	0.00	0.70	0.03	0.58	0.01	57.00	0.00	55.00	0.00	5.00	0.00	1.00	0.00	36.00	0.00	0.51	0.00	0.98	0.00	0.67	0.00
	T'_3	61.00	2.00	57.00	1.00	17.00	-3.00	14.00	-2.00	5.00	2.00	0.52	0.00	0.81	0.03	0.63	0.01	9.00	0.00	7.00	0.00	2.00	0.00	1.00	0.00	135.00	0.00	0.56	0.00	0.90	0.00	0.69	0.00
	T'_4	65.00	1.00	61.00	-1.00	14.00	1.00	8.00	-1.00	6.00	0.00	0.52	0.01	0.89	0.01	0.65	0.01	32.00	0.00	31.00	0.00	25.00	0.00	26.00	0.00	40.00	0.00	0.51	0.00	0.55	0.00	0.53	0.00
	T'_5	54.00	0.00	48.00	3.00	28.00	-3.00	22.00	0.00	2.00	0.00	0.53	-0.02	0.71	0.00	0.61	-0.01	50.00	0.00	54.00	0.00	5.00	0.00	6.00	0.00	39.00	0.00	0.48	0.00	0.89	0.00	0.62	0.00
	T'_6	59.00	1.00	54.00	2.00	17.00	-1.00	15.00	-1.00	9.00	-1.00	0.52	-0.00	0.80	0.01	0.63	0.00	47.00	0.00	45.00	0.00	13.00	0.00	11.00	0.00	38.00	0.00	0.51	0.00	0.81	0.00	0.63	0.00
	T'_7	66.00	-2.00	60.00	2.00	10.00	-1.00	8.00	1.00	10.00	0.00	0.52	-0.02	0.89	-0.02	0.66	-0.02	54.00	0.00	54.00	0.00	6.00	0.00	7.00	0.00	33.00	0.00	0.50	0.00	0.89	0.00	0.64	0.00
	T'_8	50.00	0.00	52.00	-1.00	22.00	0.00	23.00	-1.00	9.00	2.00	0.50	0.01	0.68	0.01	0.58	0.01	59.00	0.00	58.00	0.00	2.00	0.00	1.00	0.00	34.00	0.00	0.50	0.00	0.98	0.00	0.67	0.00
	T'_9	43.00	-8.00	34.00	-5.00	37.00	2.00	29.00	10.00	11.00	1.00	0.56	-0.01	0.60	-0.12	0.58	-0.07	47.00	0.00	39.00	0.00	12.00	0.00	8.00	0.00	48.00	0.00	0.55	0.00	0.85	0.00	0.67	0.00
	avg	59.65	-0.60	56.20	0.60	15.80	-0.45	14.05	0.60	8.30	-0.15	0.52	-0.01	0.81	-0.01	0.63	-0.01	51.15	0.00	50.20	0.00	7.20	0.00	6.00	0.00	39.45	0.00	0.51	0.00	0.89	0.00	0.65	0.00
Zero Shot CoT	T_0	62.00	3.00	59.00	1.00	15.00	0.00	13.00	-3.00	5.00	-1.00	0.51	0.01	0.83	0.04	0.63	0.02	8.00	0.00	17.00	0.00	12.00	0.00	13.00	0.00	104.00	0.00	0.32	0.00	0.38	0.00	0.35	0.00
	T_1	49.00	5.00	51.00	2.00	19.00	0.00	23.00	-3.00	12.00	-4.00	0.49	0.01	0.68	0.05	0.57	0.03	12.00	0.00	19.00	0.00	6.00	0.00	7.00	0.00	110.00	0.00	0.39	0.00	0.63	0.00	0.48	0.00
	T_2	52.00	6.00	54.00	-1.00	19.00	2.00	22.00	-4.00	7.00	-3.00	0.49	0.03	0.70	0.06	0.58	0.04	19.00	0.00	11.00	0.00	5.00	0.00	4.00	0.00	115.00	0.00	0.63	0.00	0.83	0.00	0.72	0.00
	T_3	66.00	0.00	59.00	6.00	15.00	-5.00	10.00	0.00	4.00	-1.00	0.53	-0.02	0.87	0.00	0.66	-0.02	20.00	0.00	15.00	0.00	0.00	0.00	2.00	0.00	117.00	0.00	0.57	0.00	0.91	0.00	0.70	0.00
	T_4	50.00	3.00	52.00	1.00	18.00	0.00	25.00	-2.00	9.00	-2.00	0.49	0.01	0.67	0.03	0.56	0.02	10.00	0.00	11.00	0.00	6.00	0.00	7.00	0.00	120.00	0.00	0.48	0.00	0.59	0.00	0.53	0.00
	T_5	58.00	2.00	53.00	6.00	19.00	-3.00	18.00	-1.00	6.00	-4.00	0.52	-0.02	0.76	0.02	0.62	-0.01	12.00	0.00	14.00	0.00	6.00	0.00	0.00	0.00	122.00	0.00	0.46	0.00	1.00	0.00	0.63	0.00
	T_6	62.00	0.00	59.00	4.00	16.00	-4.00	14.00	0.00	3.00	0.00	0.51	-0.02	0.82	0.00	0.63	-0.01	10.00	0.00	17.00	0.00	2.00	0.00	0.00	0.00	125.00	0.00	0.37	0.00	1.00	0.00	0.54	0.00
	T_7	64.00	1.00	62.00	2.00	12.00	-1.00	9.00	0.00	7.00	-2.00	0.51	-0.00	0.88	0.00	0.64	-0.00	18.00	0.00	14.00	0.00	6.00	0.00	4.00	0.00	112.00	0.00	0.56	0.00	0.82	0.00	0.67	0.00
	T_8	58.00	1.00	55.00	1.00	18.00	-1.00	17.00	-1.00	6.00	0.00	0.51	-0.00	0.77	0.01	0.62	0.00	19.00	0.00	24.00	0.00	0.00	0.00	2.00	0.00	109.00	0.00	0.44	0.00	0.90	0.00	0.59	0.00
	T_9	53.00	1.00	49.00	-1.00	23.00	2.00	22.00	0.00	7.00	-2.00	0.52	0.01	0.71	0.00	0.60	0.01	13.00	0.00	14.00	0.00	4.00	0.00	1.00	0.00	122.00	0.00	0.48	0.00	0.93	0.00	0.63	0.00
	T'_0	68.00	4.00	61.00	6.00	15.00	-7.00	6.00	-1.00	4.00	-2.00	0.53	-0.01	0.92	0.02	0.67	-0.00	21.00	0.00	19.00	0.00	19.00	0.00	16.00	0.00	79.00	0.00	0.52	0.00	0.57	0.00	0.55	0.00
	T'_1	53.00	4.00	54.00	9.00	19.00	-8.00	23.00	-4.00	5.00	-1.00	0.50	-0.02	0.70	0.05	0.58	0.00	16.00	0.00	17.00	0.00	23.00	0.00	13.00	0.00	85.00	0.00	0.48	0.00	0.55	0.00	0.52	0.00
	T'_2	38.00	-1.00	30.00	-2.00	41.00	2.00	37.00	0.00	8.00	1.00	0.56	0.01	0.51	-0.01	0.53	0.00	22.00	0.00	18.00	0.00	10.00	0.00	9.00	0.00	95.00	0.00	0.55	0.00	0.71	0.00	0.62	0.00
	T'_3	52.00	5.00	54.00	-1.00	22.00	1.00	22.00	-2.00	4.00	-3.00	0.49	0.03	0.70	0.04	0.58	0.03	6.00	0.00	9.00	0.00	2.00	0.00	5.00	0.00	132.00	0.00	0.40	0.00	0.55	0.00	0.46	0.00
	T'_4	70.00	0.00	65.00	1.00	8.00	-1.00	4.00	0.00	7.00	0.00	0.52	-0.00	0.95	0.00	0.67	-0.00	21.00	0.00	13.00	0.00	18.00	0.00	15.00	0.00	87.00	0.00	0.62	0.00	0.58	0.00	0.60	0.00
	T'_5	50.00	3.00	50.00	-4.00	26.00	3.00	23.00	1.00	5.00	-3.00	0.50	0.04	0.68	0.00	0.58	0.02	11.00	0.00	12.00	0.00	19.00	0.00	10.00	0.00	102.00	0.00	0.48	0.00	0.52	0.00	0.50	0.00
	T'_6	42.00	6.00	40.00	3.00	31.00	0.00	31.00	-3.00	10.00	-6.00	0.51	0.02	0.58	0.06	0.54	0.03	14.00	0.00	11.00	0.00	19.00	0.00	21.00	0.00	89.00	0.00	0.56	0.00	0.40	0.00	0.47	0.00
	T'_7	73.00	2.00	70.00	2.00	6.00	-3.00	4.00	-2.00	1.00	1.00	0.51	-0.00	0.95	0.03	0.66	0.01	14.00	0.00	17.00	0.00	14.00	0.00	16.00	0.00	93.00	0.00	0.45	0.00	0.47	0.00	0.46	0.00
	T'_8	64.00	-4.00	55.00	2.00	20.00	-4.00	11.00	4.00	4.00	2.00	0.54	-0.02	0.85	-0.05	0.66	-0.03	29.00	0.00	27.00	0.00	9.00	0.00	8.00	0.00	81.00	0.00	0.52	0.00	0.78	0.00	0.62	0.00
	T'_9	40.00	2.00	37.00	-1.00	36.00	2.00	35.00	-2.00	6.00	-1.00	0.52	0.02	0.53	0.03	0.53	0.02	15.00	0.00	10.00	0.00	13.00	0.00	11.00	0.00	105.00	0.00	0.60	0.00	0.58	0.00	0.59	0.00
	avg	56.20	2.15	53.45	1.80	19.90	-1.25	18.45	-1.15	6.00	-1.55	0.51	0.00	0.75	0.02	0.61	0.01	15.50	0.00	15.45	0.00	9.65	0.00	8.20	0.00	105.20	0.00	0.49	0.00	0.68	0.00	0.56	0.00
Overall AVG		57.92	0.77	54.82	1.20	17.85	-0.85	16.25	-0.27	7.15	-0.85	0.52	-0.00	0.78	0.00	0.62	0.00	33.33	0.00	32.83	0.00	8.43	0.00	7.10	0.00	72.32	0.00	0.50	0.00	0.78	0.00	0.60	0.00

Table 1: Performance of the two models and two prompting techniques for each task executed ten times and aggregated with a majority voting strategy. The ΔS indicates the difference with the performance from the single-run execution (from Table 3).

Table 2: Average cosine similarity of all unique pairs of the ten responses resulting from two models and two prompting techniques for each task. The Δ indicates the difference between a task and its corresponding inverted task.

		GPT-3.5										Llama-2									
	Task	cos(TP)	Δ	cos(FP)	Δ	cos(TN)	Δ	cos(FN)	Δ	cos(na)	Δ	cos(TP)	Δ	cos(FP)	Δ	cos(TN)	Δ	cos(FN)	Δ	cos(NA)	Δ
Zero Shot	T_0	0.71	/	0.72	/	0.55	/	0.44	/	0.70	/	0.99	/	1.00	/	1.00	/	1.00	/	0.83	/
	T_1	0.71	/	0.70	/	0.69	/	0.70	/	0.73	/	1.00	/	1.00	/	1.00	/	1.00	/	0.88	/
	T_2	0.75	/	0.76	/	0.71	/	0.73	/	0.71	/	1.00	/	1.00	/	1.00	/	1.00	/	0.97	/
	T_3	0.77	/	0.77	/	0.75	/	0.72	/	0.69	/	1.00	/	1.00	/	1.00	/	1.00	/	0.97	/
	T_4	0.70	/	0.70	/	0.70	/	0.69	/	0.68	/	1.00	/	1.00	/	1.00	/	1.00	/	0.88	/
	T_5	0.74	/	0.74	/	0.70	/	0.69	/	0.81	/	1.00	/	1.00	/	1.00	/	1.00	/	0.95	/
	T_6	0.74	/	0.75	/	0.77	/	0.72	/	0.80	/	1.00	/	1.00	/	1.00	/	1.00	/	1.00	/
	T_7	0.71	/	0.71	/	0.70	/	0.72	/	0.74	/	1.00	/	1.00	/	1.00	/	1.00	/	0.85	/
	T_8	0.72	/	0.69	/	0.70	/	0.69	/	0.72	/	1.00	/	1.00	/	1.00	/	1.00	/	0.92	/
	T_9	0.75	/	0.75	/	0.69	/	0.69	/	0.69	/	1.00	/	1.00	/	1.00	/	1.00	/	0.90	/
	T'_0	0.67	(-0.04)	0.68	(-0.04)	0.61	(0.05)	0.58	(0.14)	0.58	(-0.12)	1.00	(0.01)	1.00	(0.0)	1.00	(0.0)	0.00	(-1.0)	0.58	(-0.25)
	T'_1	0.72	(0.01)	0.72	(0.01)	0.64	(-0.04)	0.65	(-0.05)	0.72	(-0.01)	1.00	(0.0)	1.00	(0.0)	1.00	(0.0)	1.00	(-0.0)	0.67	(-0.22)
	T'_2	0.72	(-0.03)	0.73	(-0.03)	0.70	(-0.01)	0.71	(-0.02)	0.71	(0.01)	1.00	(0.0)	1.00	(0.0)	1.00	(0.0)	1.00	(0.0)	0.86	(-0.11)
	T'_3	0.73	(-0.03)	0.74	(-0.03)	0.73	(-0.02)	0.74	(0.03)	0.74	(0.05)	1.00	(0.0)	1.00	(0.0)	1.00	(0.0)	1.00	(0.0)	1.00	(0.03)
	T'_4	0.72	(0.02)	0.68	(-0.02)	0.64	(-0.06)	0.62	(-0.08)	0.72	(0.04)	1.00	(0.0)	1.00	(-0.0)	1.00	(-0.0)	1.00	(-0.0)	0.82	(-0.06)
	T'_5	0.66	(-0.08)	0.69	(-0.05)	0.66	(-0.04)	0.66	(-0.03)	0.58	(-0.23)	1.00	(0.0)	1.00	(0.0)	1.00	(-0.0)	1.00	(0.0)	0.72	(-0.23)
	T'_6	0.71	(-0.04)	0.71	(-0.04)	0.69	(-0.08)	0.69	(-0.02)	0.71	(-0.09)	1.00	(0.0)	1.00	(0.0)	1.00	(0.0)	1.00	(0.0)	0.82	(-0.18)
	T'_7	0.71	(-0.01)	0.70	(-0.01)	0.68	(-0.02)	0.65	(-0.07)	0.73	(-0.01)	1.00	(0.0)	1.00	(0.0)	1.00	(-0.0)	1.00	(-0.0)	0.88	(0.03)
	T'_8	0.71	(-0.01)	0.71	(0.03)	0.70	(-0.01)	0.72	(0.04)	0.74	(0.02)	1.00	(0.0)	1.00	(0.0)	1.00	(0.0)	1.00	(0.0)	0.85	(-0.07)
	T'_9	0.66	(-0.09)	0.67	(-0.08)	0.66	(-0.03)	0.62	(-0.06)	0.72	(0.03)	1.00	(0.0)	1.00	(0.0)	1.00	(0.0)	1.00	(0.0)	0.79	(-0.1)
	avg	0.72	/	0.72	/	0.68	/	0.67	/	0.71	/	1.00	/	1.00	/	1.00	/	0.95	/	0.86	/
Zero Shot CoT	T_0	0.74	/	0.73	/	0.70	/	0.72	/	0.78	/	1.00	/	1.00	/	1.00	/	1.00	/	1.00	/
	T_1	0.73	/	0.74	/	0.73	/	0.75	/	0.80	/	1.00	/	1.00	/	1.00	/	1.00	/	1.00	/
	T_2	0.75	/	0.75	/	0.77	/	0.75	/	0.73	/	1.00	/	1.00	/	1.00	/	1.00	/	1.00	/
	T_3	0.74	/	0.76	/	0.73	/	0.76	/	0.72	/	1.00	/	1.00	/	0.00	/	1.00	/	1.00	/
	T_4	0.74	/	0.72	/	0.75	/	0.73	/	0.77	/	1.00	/	1.00	/	1.00	/	1.00	/	1.00	/
	T_5	0.73	/	0.73	/	0.75	/	0.75	/	0.70	/	1.00	/	1.00	/	1.00	/	0.00	/	1.00	/
	T_6	0.74	/	0.73	/	0.75	/	0.75	/	0.65	/	1.00	/	1.00	/	1.00	/	0.00	/	1.00	/
	T_7	0.74	/	0.74	/	0.75	/	0.70	/	0.75	/	1.00	/	1.00	/	1.00	/	1.00	/	1.00	/
	T_8	0.76	/	0.74	/	0.77	/	0.77	/	0.72	/	1.00	/	1.00	/	0.00	/	1.00	/	1.00	/
	T_9	0.74	/	0.74	/	0.75	/	0.75	/	0.76	/	1.00	/	1.00	/	1.00	/	1.00	/	1.00	/
	T'_0	0.72	(-0.02)	0.73	(-0.01)	0.72	(0.02)	0.69	(-0.03)	0.79	(0.01)	1.00	(-0.0)	1.00	(0.0)	1.00	(0.0)	1.00	(0.0)	1.00	(0.0)
	T'_1	0.75	(0.02)	0.74	(0.0)	0.71	(-0.02)	0.72	(-0.03)	0.84	(0.03)	1.00	(0.0)	1.00	(0.0)	1.00	(0.0)	1.00	(0.0)	1.00	(0.0)
	T'_2	0.76	(0.02)	0.76	(0.01)	0.77	(-0.0)	0.75	(-0.0)	0.75	(0.02)	1.00	(0.0)	1.00	(0.0)	1.00	(0.0)	1.00	(0.0)	1.00	(-0.0)
	T'_3	0.78	(0.03)	0.78	(0.02)	0.77	(0.04)	0.76	(0.01)	0.79	(0.07)	1.00	(0.0)	1.00	(0.0)	1.00	(1.0)	1.00	(0.0)	1.00	(0.0)
	T'_4	0.72	(-0.02)	0.72	(0.0)	0.71	(-0.04)	0.70	(-0.03)	0.80	(0.03)	1.00	(0.0)	1.00	(0.0)	1.00	(0.0)	1.00	(-0.0)	1.00	(0.0)
	T'_5	0.69	(-0.04)	0.72	(-0.02)	0.73	(-0.02)	0.72	(-0.03)	0.66	(-0.04)	1.00	(0.0)	1.00	(0.0)	1.00	(0.0)	1.00	(1.0)	1.00	(0.0)
	T'_6	0.75	(0.01)	0.74	(0.0)	0.76	(0.01)	0.74	(-0.01)	0.70	(0.05)	1.00	(0.0)	1.00	(0.0)	1.00	(0.0)	1.00	(1.0)	1.00	(0.0)
	T'_7	0.74	(-0.01)	0.73	(-0.02)	0.73	(-0.02)	0.70	(0.0)	0.63	(-0.12)	1.00	(0.0)	1.00	(0.0)	1.00	(0.0)	1.00	(0.0)	1.00	(0.0)
	T'_8	0.75	(-0.01)	0.73	(-0.01)	0.76	(-0.01)	0.77	(0.0)	0.84	(0.11)	1.00	(0.0)	1.00	(0.0)	1.00	(1.0)	1.00	(-0.0)	1.00	(0.0)
	T'_9	0.73	(-0.0)	0.70	(-0.04)	0.71	(-0.03)	0.72	(-0.03)	0.68	(-0.08)	1.00	(0.0)	1.00	(0.0)	1.00	(0.0)	1.00	(0.0)	1.00	(0.0)
	avg	0.74	/	0.74	/	0.74	/	0.74	/	0.74	/	1.00	/	1.00	/	0.90	/	0.90	/	1.00	/
Overall AVG		0.73	/	0.73	/	0.71	/	0.70	/	0.72	/	1.00	/	1.00	/	0.95	/	0.92	/	0.93	/

Table 3: Number of punctuation characters (`punc`) in the responses generated by the two models and two prompting techniques for each task. The Δ indicates the difference between a task and its corresponding inverted task.

		GPT-3.5								Llama-2							
	Task	punc(TP)	(Δ)	punc(FP)	(Δ)	punc(TN)	(Δ)	punc(FN)	(Δ)	punc(NA)	(Δ)	punc(TP)	(Δ)	punc(FP)	(Δ)	punc(TN)	(Δ)	punc(FN)	(Δ)	punc(NA)	(Δ)
Zero Shot	T_0	32.60	/	31.44	/	20.57	/	22.00	/	22.12	/	45.24	/	42.53	/	18.44	/	21.00	/	67.96	/
	T_1	40.24	/	38.37	/	28.73	/	21.56	/	34.38	/	31.69	/	31.00	/	9.62	/	23.30	/	53.54	/
	T_2	51.70	/	55.11	/	34.78	/	28.70	/	24.29	/	31.98	/	30.25	/	33.00	/	26.50	/	59.76	/
	T_3	51.87	/	55.94	/	23.25	/	30.89	/	24.83	/	31.68	/	30.44	/	27.50	/	30.67	/	56.97	/
	T_4	34.91	/	38.17	/	19.26	/	12.58	/	23.42	/	33.88	/	36.02	/	7.57	/	8.36	/	73.50	/
	T_5	46.39	/	50.39	/	25.33	/	30.30	/	38.20	/	33.69	/	32.51	/	33.14	/	43.50	/	45.54	/
	T_6	48.75	/	53.96	/	25.88	/	32.33	/	17.00	/	26.60	/	27.49	/	28.33	/	36.00	/	56.35	/
	T_7	38.24	/	39.61	/	16.38	/	17.75	/	23.38	/	40.55	/	47.92	/	43.00	/	28.25	/	65.69	/
	T_8	36.38	/	37.24	/	31.07	/	23.83	/	26.10	/	36.59	/	45.51	/	59.60	/	56.00	/	60.80	/
	T_9	56.03	/	53.75	/	27.89	/	25.36	/	29.67	/	36.86	/	31.80	/	27.00	/	35.25	/	59.58	/
	T_0'	36.82	(4.22)	38.53	(7.09)	21.83	(1.26)	21.09	(-0.91)	19.46	(-2.66)	50.53	(5.3)	56.63	(14.1)	92.00	(73.56)	0.00	(-21.0)	70.77	(2.82)
	T_1'	42.75	(2.52)	43.22	(4.86)	28.24	(-0.49)	26.82	(5.26)	33.79	(-0.59)	35.97	(4.28)	34.25	(3.25)	28.00	(18.38)	35.00	(11.7)	68.87	(15.33)
	T_2'	49.37	(-2.33)	46.02	(-9.09)	47.70	(12.92)	42.73	(14.03)	34.00	(9.71)	37.39	(5.4)	36.20	(5.95)	24.20	(-8.8)	33.00	(6.5)	61.72	(1.96)
	T_3'	46.34	(-5.52)	48.96	(-6.98)	25.59	(2.34)	27.36	(-3.53)	14.60	(-10.23)	35.67	(3.98)	39.29	(8.84)	38.00	(10.5)	2.00	(-28.67)	65.47	(8.49)
	T_4'	34.26	(-0.65)	33.52	(-4.65)	20.29	(1.03)	16.62	(4.05)	20.17	(-3.25)	48.00	(14.12)	42.29	(6.27)	46.12	(38.55)	41.19	(32.84)	77.95	(4.45)
	T_5'	50.76	(4.37)	45.29	(-5.1)	27.79	(2.45)	28.45	(-1.85)	21.00	(-17.2)	39.78	(6.09)	46.13	(13.62)	5.80	(-27.34)	10.67	(-32.83)	55.00	(9.46)
	T_6'	48.27	(-0.48)	49.28	(-4.68)	31.00	(5.12)	32.13	(-0.2)	26.44	(9.44)	35.85	(9.25)	32.40	(4.91)	34.85	(6.51)	25.82	(-10.18)	62.74	(6.39)
	T_7'	39.55	(1.3)	35.42	(-4.19)	26.90	(10.52)	23.50	(5.75)	24.00	(0.62)	63.09	(22.54)	59.13	(11.21)	32.67	(-10.33)	71.43	(43.18)	64.61	(-1.09)
	T_8'	42.22	(5.84)	48.14	(10.9)	26.59	(-4.48)	29.61	(5.78)	28.11	(2.01)	41.75	(5.16)	42.93	(-2.58)	59.00	(-0.6)	65.00	(9.0)	73.35	(12.55)
	T_9'	47.72	(-8.31)	38.18	(-15.57)	15.95	(-11.94)	16.41	(-8.95)	21.64	(-8.03)	44.94	(8.08)	46.15	(14.35)	31.75	(4.75)	26.88	(-8.38)	58.69	(-0.9)
	avg	43.76	/	44.03	/	26.25	/	25.50	/	25.33	/	39.09	/	39.54	/	33.98	/	30.99	/	62.94	/
Zero Shot CoT	T_0	62.76	/	68.31	/	55.33	/	63.08	/	57.00	/	237.62	/	288.59	/	242.58	/	377.54	/	406.59	/
	T_1	71.18	/	70.39	/	63.00	/	64.57	/	63.67	/	260.33	/	234.26	/	208.17	/	284.43	/	377.92	/
	T_2	71.94	/	80.50	/	62.37	/	67.27	/	58.43	/	381.32	/	277.00	/	290.60	/	153.00	/	348.76	/
	T_3	70.77	/	71.76	/	62.13	/	59.80	/	55.25	/	279.15	/	223.33	/	0.00	/	195.00	/	374.99	/
	T_4	61.66	/	64.69	/	52.06	/	52.40	/	55.22	/	268.50	/	249.91	/	349.33	/	247.29	/	363.34	/
	T_5	72.05	/	76.08	/	61.58	/	62.56	/	53.17	/	246.17	/	244.71	/	362.83	/	0.00	/	366.71	/
	T_6	74.44	/	72.42	/	63.06	/	65.71	/	69.33	/	311.00	/	314.00	/	639.00	/	0.00	/	341.90	/
	T_7	65.34	/	69.08	/	49.42	/	63.00	/	52.43	/	295.39	/	335.64	/	184.00	/	180.50	/	373.47	/
	T_8	65.24	/	68.71	/	62.61	/	62.00	/	61.33	/	321.79	/	251.33	/	0.00	/	230.00	/	373.55	/
	T_9	77.51	/	73.78	/	58.70	/	61.77	/	57.71	/	285.00	/	255.64	/	224.25	/	654.00	/	374.91	/
	T_0'	61.09	(-1.67)	60.75	(-7.55)	65.87	(10.53)	65.50	(2.42)	48.25	(-8.75)	339.14	(101.52)	243.05	(-45.54)	346.58	(104.0)	333.69	(-43.85)	392.33	(-14.26)
	T_1'	61.25	(-9.94)	66.59	(-3.8)	65.00	(2.0)	55.30	(-9.26)	54.40	(-9.27)	255.56	(-4.77)	298.88	(64.62)	281.17	(73.01)	436.46	(152.03)	381.12	(3.2)
	T_2'	74.61	(2.66)	73.57	(-6.93)	65.10	(2.73)	70.32	(3.05)	75.50	(17.07)	352.95	(-28.36)	309.67	(32.67)	226.70	(-63.9)	373.22	(220.22)	371.52	(22.76)
	T_3'	73.27	(2.5)	75.72	(3.96)	61.23	(-0.91)	67.55	(7.75)	71.25	(16.0)	305.17	(26.02)	335.11	(111.78)	621.50	(621.5)	411.60	(216.6)	373.70	(-1.29)
	T_4'	61.07	(-0.59)	59.86	(-4.83)	60.12	(8.07)	51.25	(-1.15)	52.71	(-2.51)	351.14	(82.64)	299.77	(49.86)	279.39	(-69.94)	439.53	(192.25)	354.21	(-9.13)
	T_5'	64.04	(-7.98)	63.68	(-12.4)	55.19	(-6.39)	61.04	(-1.51)	53.60	(0.43)	313.09	(66.92)	369.50	(124.79)	351.68	(-111.15)	516.30	(516.3)	377.04	(10.33)
	T_6'	77.74	(3.3)	68.18	(-4.25)	65.52	(2.45)	61.19	(-4.52)	65.10	(-4.23)	301.93	(-9.07)	290.27	(-23.73)	251.84	(-387.16)	482.38	(482.38)	376.96	(35.05)
	T_7'	59.45	(-5.89)	59.04	(-10.04)	62.33	(12.92)	51.75	(-11.25)	18.00	(-34.43)	312.86	(17.47)	271.35	(-64.29)	237.00	(53.0)	357.69	(177.19)	384.48	(11.01)
	T_8'	63.36	(-1.88)	64.00	(-4.71)	65.30	(2.69)	59.27	(-2.73)	50.75	(-10.58)	380.17	(58.38)	310.33	(59.0)	209.78	(209.78)	266.50	(36.5)	364.64	(-8.91)
	T_9'	65.15	(-12.36)	63.97	(-9.8)	48.53	(-10.17)	56.89	(-4.89)	56.67	(-1.05)	280.53	(-4.47)	295.40	(39.76)	324.31	(100.06)	527.55	(-126.45)	370.79	(-4.12)
	avg	67.69	/	68.55	/	60.22	/	61.11	/	56.49	/	303.94	/	284.89	/	286.54	/	323.33	/	372.45	/
Overall AVG		55.72	/	56.29	/	43.24	/	43.30	/	40.91	/	171.52	/	162.22	/	160.26	/	177.16	/	217.70	/

Table 4: Coleman Linau score (`col`) of the responses generated by the two models and two prompting techniques for each task. The Δ indicates the difference between a task and its corresponding inverted task.

		GPT-3.5								Llama-2							
	Task	col(TP)	Δ	col(FP)	Δ	col(TN)	Δ	col(FN)	Δ	col(NA)	Δ	col(TP)	Δ	col(FP)	Δ	col(TN)	Δ	col(FN)	Δ	col(NA)	Δ
Zero Shot	T_0	12.07	/	12.31	/	8.90	/	0.00	/	9.44	/	10.04	/	9.91	/	3.35	/	5.25	/	0.25	/
	T_1	13.27	/	12.78	/	14.91	/	12.67	/	13.81	/	6.84	/	6.94	/	0.00	/	1.97	/	2.02	/
	T_2	14.96	/	15.07	/	15.11	/	14.40	/	13.38	/	6.43	/	7.58	/	9.41	/	5.01	/	4.27	/
	T_3	14.72	/	14.67	/	14.56	/	14.20	/	12.52	/	7.88	/	7.83	/	7.46	/	8.70	/	2.60	/
	T_4	12.64	/	13.13	/	6.18	/	2.62	/	12.43	/	9.62	/	8.65	/	0.00	/	0.00	/	2.15	/
	T_5	14.58	/	15.08	/	12.57	/	13.30	/	10.12	/	7.85	/	6.94	/	8.79	/	5.98	/	4.19	/
	T_6	14.71	/	14.87	/	14.52	/	14.19	/	7.62	/	7.47	/	7.93	/	9.19	/	9.62	/	3.69	/
	T_7	14.29	/	13.30	/	9.19	/	5.67	/	9.64	/	10.70	/	11.32	/	8.85	/	2.70	/	3.30	/
	T_8	14.49	/	13.67	/	13.40	/	12.74	/	12.33	/	6.73	/	8.55	/	4.03	/	7.33	/	1.44	/
	T_9	14.91	/	15.37	/	10.11	/	14.72	/	13.25	/	7.12	/	7.67	/	6.62	/	8.28	/	5.70	/
	T_0'	14.38	(2.31)	14.16	(1.85)	7.86	(-1.03)	9.31	(9.31)	10.10	(0.66)	7.87	(-2.17)	9.49	(-0.42)	15.37	(12.02)	0.00	(-5.25)	2.37	(2.12)
	T_1'	14.54	(1.27)	15.15	(2.37)	11.48	(-3.44)	13.27	(0.61)	13.23	(-0.59)	5.92	(-0.93)	6.50	(-0.43)	0.00	(0.0)	0.00	(-1.97)	2.78	(0.76)
	T_2'	15.71	(0.75)	15.63	(0.57)	14.35	(-0.76)	14.90	(0.49)	13.28	(-0.1)	7.03	(0.6)	7.22	(-0.35)	4.70	(-4.71)	6.76	(1.75)	3.12	(-1.15)
	T_3'	15.13	(0.41)	15.02	(0.35)	13.27	(-1.29)	13.48	(-0.72)	12.60	(0.08)	5.83	(-2.05)	5.37	(-2.46)	9.59	(2.13)	0.00	(-8.7)	4.73	(2.13)
	T_4'	13.77	(1.13)	12.98	(-0.15)	10.18	(4.0)	9.29	(6.67)	12.70	(0.27)	7.10	(-2.52)	6.79	(-1.86)	8.43	(8.43)	6.41	(6.41)	5.93	(3.77)
	T_5'	15.29	(0.71)	15.48	(0.4)	12.89	(0.32)	13.45	(0.15)	8.03	(-2.09)	6.34	(-1.51)	6.78	(-0.16)	1.61	(-7.19)	1.41	(-4.57)	4.81	(0.62)
	T_6'	15.28	(0.57)	15.37	(0.51)	15.23	(0.71)	15.00	(0.81)	14.46	(6.84)	7.91	(0.44)	8.22	(0.3)	7.22	(-1.97)	6.17	(-3.46)	6.15	(2.46)
	T_7'	14.89	(0.6)	14.51	(1.21)	13.75	(4.55)	14.57	(8.9)	14.65	(5.01)	10.34	(-0.36)	10.82	(-0.5)	4.87	(-3.98)	9.39	(6.69)	2.47	(-0.84)
	T_8'	15.47	(0.98)	15.21	(1.53)	12.91	(-0.49)	13.74	(1.0)	13.71	(1.38)	9.34	(2.61)	9.50	(0.95)	13.23	(9.2)	10.43	(3.1)	4.29	(2.85)
	T_9'	14.53	(-0.39)	15.16	(-0.21)	6.34	(-3.77)	7.84	(-6.89)	12.57	(-0.68)	6.71	(-0.41)	6.39	(-1.28)	5.95	(-0.66)	6.45	(-1.83)	5.40	(-0.3)
	avg	14.48	/	14.45	/	11.89	/	11.47	/	11.99	/	7.75	/	8.02	/	6.43	/	5.09	/	3.58	/
Zero Shot CoT	T_0	14.23	/	14.44	/	14.36	/	14.70	/	13.04	/	12.95	/	14.04	/	15.17	/	12.05	/	15.43	/
	T_1	14.65	/	15.11	/	14.57	/	14.92	/	14.72	/	14.50	/	13.82	/	10.90	/	14.92	/	15.61	/
	T_2	14.28	/	14.57	/	13.80	/	14.66	/	16.15	/	13.47	/	12.78	/	14.35	/	11.72	/	14.25	/
	T_3	14.29	/	14.29	/	14.85	/	14.20	/	15.55	/	12.05	/	13.13	/	0.00	/	16.62	/	14.91	/
	T_4	14.74	/	15.02	/	14.69	/	14.53	/	13.75	/	13.64	/	11.13	/	16.80	/	10.51	/	15.11	/
	T_5	14.14	/	14.45	/	14.21	/	14.85	/	15.66	/	10.94	/	12.46	/	13.49	/	0.00	/	15.21	/
	T_6	14.14	/	14.35	/	14.50	/	14.28	/	11.05	/	15.09	/	12.38	/	17.76	/	0.00	/	14.90	/
	T_7	14.85	/	14.36	/	14.38	/	14.25	/	13.63	/	13.93	/	16.15	/	10.67	/	10.46	/	15.84	/
	T_8	14.16	/	14.56	/	14.61	/	14.88	/	15.22	/	14.07	/	12.90	/	0.00	/	9.31	/	14.15	/
	T_9	14.65	/	14.33	/	14.82	/	14.35	/	13.34	/	14.06	/	11.75	/	20.74	/	31.50	/	15.44	/
	T_0'	14.77	(0.54)	14.77	(0.33)	14.93	(0.57)	14.69	(-0.02)	11.52	(-1.52)	14.10	(1.15)	14.10	(0.06)	14.09	(-1.09)	14.67	(2.61)	16.54	(1.11)
	T_1'	14.61	(-0.04)	14.96	(-0.15)	14.36	(-0.21)	14.44	(-0.48)	12.00	(-2.71)	12.86	(-1.65)	11.93	(-1.89)	15.44	(4.53)	13.46	(-1.45)	15.51	(-0.1)
	T_2'	14.32	(0.04)	14.73	(0.15)	14.74	(0.94)	14.50	(-0.16)	13.66	(-2.48)	12.56	(-0.91)	13.89	(1.11)	14.18	(-0.17)	14.14	(2.42)	15.22	(0.97)
	T_3'	14.52	(0.22)	14.93	(0.65)	14.94	(0.09)	15.04	(0.84)	12.77	(-2.78)	13.45	(1.39)	17.11	(3.98)	19.70	(19.7)	14.61	(-2.01)	16.65	(1.74)
	T_4'	14.99	(0.25)	14.95	(-0.08)	14.69	(-0.0)	14.19	(-0.34)	13.41	(-0.34)	14.89	(1.25)	14.16	(3.03)	14.18	(-2.62)	16.17	(5.66)	13.42	(-1.69)
	T_5'	14.69	(0.55)	15.18	(0.73)	15.23	(1.02)	14.31	(-0.54)	11.94	(-3.71)	13.51	(2.56)	13.15	(0.69)	15.54	(2.06)	14.09	(14.09)	16.62	(1.41)
	T_6'	14.71	(0.57)	14.15	(-0.19)	14.98	(0.48)	14.67	(0.38)	15.86	(4.81)	15.96	(0.87)	14.67	(2.29)	14.03	(-3.73)	16.16	(16.16)	15.52	(0.62)
	T_7'	14.43	(-0.42)	14.55	(0.2)	15.95	(1.57)	15.15	(0.89)	0.00	(-13.63)	13.26	(-0.66)	12.64	(-3.51)	12.56	(1.88)	13.45	(2.99)	15.85	(0.01)
	T_8'	14.56	(0.4)	14.84	(0.28)	14.66	(0.05)	14.27	(-0.61)	14.83	(-0.4)	14.05	(-0.02)	11.85	(-1.06)	14.05	(14.05)	15.64	(6.34)	14.15	(-0.0)
	T_9'	14.87	(0.22)	14.93	(0.6)	14.16	(-0.65)	14.28	(-0.07)	12.88	(-0.46)	14.58	(0.52)	12.24	(0.49)	16.94	(-3.8)	14.77	(-16.73)	16.91	(1.47)
	avg	14.53	/	14.67	/	14.67	/	14.56	/	13.05	/	13.70	/	13.31	/	13.53	/	13.21	/	15.36	/
Overall AVG		14.50	/	14.56	/	13.28	/	13.02	/	12.52	/	10.72	/	10.66	/	9.98	/	9.15	/	9.47	/

Table 5: Ari score (`ari`) of the responses generated by the two models and two prompting techniques for each task. The **Δ** indicates the difference between a task and its corresponding inverted task.

		GPT-3.5										Llama-2									
	Task	ari(TP)	Δ	ari(FP)	Δ	ari(TN)	Δ	ari(FN)	Δ	ari(NA)	Δ	ari(TP)	Δ	ari(FP)	Δ	ari(TN)	Δ	ari(FN)	Δ	ari(NA)	Δ
Zero Shot	T_0	9.93	/	10.03	/	9.32	/	0.00	/	9.27	/	9.72	/	9.88	/	5.25	/	6.47	/	3.97	/
	T_1	11.49	/	10.90	/	15.09	/	12.57	/	13.51	/	7.27	/	7.25	/	0.00	/	1.82	/	11.11	/
	T_2	12.56	/	12.69	/	16.15	/	14.50	/	13.30	/	7.10	/	7.08	/	10.26	/	4.93	/	7.04	/
	T_3	12.22	/	12.00	/	14.89	/	13.22	/	11.36	/	7.68	/	7.13	/	6.90	/	6.10	/	5.23	/
	T_4	10.47	/	11.02	/	6.55	/	2.83	/	12.04	/	9.88	/	8.45	/	0.00	/	0.00	/	13.19	/
	T_5	12.10	/	12.33	/	13.18	/	13.09	/	8.36	/	8.11	/	6.64	/	9.33	/	4.65	/	6.50	/
	T_6	12.38	/	12.21	/	14.50	/	13.71	/	8.02	/	6.59	/	7.65	/	7.59	/	6.98	/	5.94	/
	T_7	11.78	/	11.03	/	9.30	/	5.34	/	8.64	/	10.46	/	12.06	/	9.23	/	2.60	/	12.34	/
	T_8	13.50	/	12.58	/	14.06	/	12.76	/	12.88	/	7.43	/	8.99	/	3.47	/	5.67	/	2.31	/
	T_9	12.66	/	12.97	/	10.42	/	14.92	/	13.31	/	6.43	/	6.59	/	7.65	/	8.54	/	11.33	/
	T_0^i	12.58	(2.65)	12.32	(2.3)	8.51	(-0.8)	10.22	(10.22)	9.70	(0.43)	7.71	(-2.01)	10.06	(0.17)	15.44	(10.19)	0.00	(-6.47)	3.78	(-0.19)
	T_1^i	12.91	(1.42)	13.16	(2.26)	10.91	(-4.18)	12.65	(0.08)	12.66	(-0.84)	6.07	(-1.19)	6.59	(-0.66)	0.00	(0.0)	0.00	(-1.82)	5.54	(-5.57)
	T_2^i	13.67	(1.12)	14.06	(1.38)	13.57	(-2.58)	13.41	(-1.09)	11.73	(-1.56)	10.85	(3.76)	7.69	(0.61)	2.67	(-7.6)	4.21	(-0.72)	4.64	(-2.41)
	T_3^i	12.66	(0.44)	12.63	(0.64)	13.94	(-0.96)	14.54	(1.31)	12.93	(1.58)	5.00	(-2.68)	4.99	(-2.15)	18.62	(11.71)	0.00	(-6.1)	5.92	(0.69)
	T_4^i	12.26	(1.79)	11.47	(0.45)	11.07	(4.52)	9.49	(6.66)	12.63	(0.59)	5.89	(-3.99)	6.00	(-2.45)	7.57	(7.57)	5.41	(5.41)	9.01	(-4.18)
	T_5^i	13.82	(1.71)	13.86	(1.53)	12.48	(-0.7)	13.22	(0.13)	6.55	(-1.81)	7.52	(-0.59)	6.62	(-0.02)	1.25	(-8.08)	0.88	(-3.77)	7.30	(0.79)
	T_6^i	12.96	(0.58)	12.82	(0.61)	15.27	(0.77)	14.34	(0.64)	13.16	(5.14)	10.76	(4.17)	9.80	(2.14)	11.90	(4.32)	13.14	(6.16)	9.08	(3.14)
	T_7^i	13.08	(1.3)	12.50	(1.47)	13.44	(4.14)	15.61	(10.28)	13.46	(4.83)	10.62	(0.15)	10.46	(-1.61)	4.78	(-4.45)	11.51	(8.91)	3.26	(-9.08)
	T_8^i	13.71	(0.21)	13.05	(0.48)	12.63	(-1.44)	13.04	(0.29)	12.92	(0.04)	9.26	(1.83)	9.19	(0.19)	14.12	(10.66)	6.55	(0.88)	6.81	(4.5)
	T_9^i	13.26	(0.6)	13.20	(0.23)	6.68	(-3.75)	7.76	(-7.17)	12.08	(-1.23)	6.02	(-0.4)	5.62	(-0.97)	6.94	(-0.72)	5.08	(-3.46)	7.97	(-3.37)
	avg	12.50	/	12.34	/	12.10	/	11.36	/	11.43	/	8.02	/	7.94	/	7.15	/	4.73	/	7.11	/
Zero Shot CoT	T_0	11.65	/	11.93	/	12.85	/	12.59	/	11.59	/	12.11	/	13.66	/	16.87	/	11.58	/	18.83	/
	T_1	12.03	/	12.49	/	13.00	/	12.73	/	13.68	/	14.29	/	13.90	/	10.57	/	16.44	/	21.20	/
	T_2	11.61	/	11.89	/	12.15	/	12.64	/	14.46	/	18.85	/	12.72	/	11.18	/	10.22	/	16.19	/
	T_3	11.16	/	11.20	/	12.41	/	11.67	/	13.50	/	12.49	/	13.08	/	0.00	/	17.28	/	18.58	/
	T_4	12.20	/	12.49	/	13.19	/	12.79	/	13.52	/	15.19	/	9.82	/	16.05	/	9.19	/	17.57	/
	T_5	11.19	/	11.43	/	11.58	/	12.16	/	13.86	/	10.67	/	12.34	/	17.99	/	0.00	/	17.49	/
	T_6	11.20	/	11.26	/	11.81	/	11.92	/	9.03	/	16.75	/	12.39	/	28.05	/	0.00	/	18.35	/
	T_7	11.98	/	11.42	/	13.22	/	11.77	/	12.41	/	15.28	/	17.68	/	11.05	/	8.46	/	19.44	/
	T_8	11.73	/	12.48	/	12.80	/	12.91	/	13.14	/	13.01	/	12.40	/	0.00	/	7.41	/	17.66	/
	T_9	12.07	/	11.57	/	12.66	/	12.22	/	12.03	/	15.39	/	13.29	/	21.81	/	29.90	/	20.54	/
	T_0^i	12.39	(0.73)	12.28	(0.35)	12.88	(0.03)	12.88	(0.29)	11.35	(-0.24)	17.60	(5.49)	14.59	(0.94)	16.84	(-0.03)	17.83	(6.25)	18.71	(-0.12)
	T_1^i	11.99	(-0.04)	12.82	(0.33)	13.05	(0.04)	12.38	(-0.35)	10.76	(-2.92)	13.54	(-0.75)	12.81	(-1.1)	16.20	(5.63)	15.78	(-0.65)	19.37	(-1.83)
	T_2^i	11.47	(-0.14)	12.08	(0.19)	12.23	(0.08)	12.35	(-0.29)	11.35	(-3.12)	13.41	(-5.44)	14.77	(2.04)	12.90	(1.71)	14.79	(4.57)	18.15	(1.95)
	T_3^i	11.79	(0.63)	12.14	(0.94)	13.02	(0.61)	13.03	(1.37)	11.39	(-2.12)	14.85	(2.36)	17.42	(4.34)	27.00	(27.0)	20.25	(2.96)	21.62	(3.04)
	T_4^i	12.55	(0.35)	12.82	(0.32)	13.70	(0.51)	13.69	(0.91)	11.80	(-1.73)	16.15	(0.96)	13.51	(3.68)	13.63	(-2.42)	17.78	(8.59)	14.87	(-2.7)
	T_5^i	12.52	(1.33)	13.16	(1.74)	13.01	(1.44)	12.18	(0.02)	9.64	(-4.23)	13.65	(2.97)	14.68	(2.34)	15.22	(-2.77)	19.69	(19.69)	20.74	(3.25)
	T_6^i	12.18	(0.98)	11.78	(0.52)	12.57	(0.76)	12.33	(0.4)	14.06	(5.03)	14.18	(-2.56)	15.00	(2.6)	12.33	(-15.72)	18.95	(18.95)	17.67	(-0.68)
	T_7^i	12.05	(0.07)	12.11	(0.7)	14.47	(1.25)	14.59	(2.82)	0.00	(-12.41)	14.38	(-0.9)	11.54	(-6.14)	12.83	(1.78)	24.28	(15.82)	19.97	(0.53)
	T_8^i	12.13	(0.4)	12.39	(-0.09)	12.52	(-0.28)	13.10	(0.19)	14.04	(0.9)	19.36	(6.35)	11.11	(-1.29)	12.61	(12.61)	14.34	(6.93)	15.28	(-2.38)
	T_9^i	12.79	(0.72)	12.86	(1.29)	12.23	(-0.43)	12.48	(0.26)	12.05	(0.02)	13.85	(-1.54)	12.75	(-0.54)	19.44	(-2.37)	23.78	(-6.12)	21.94	(1.4)
	avg	11.93	/	12.13	/	12.77	/	12.62	/	11.68	/	14.75	/	13.47	/	14.63	/	14.90	/	18.71	/
	Overall AVG	12.22	/	12.24	/	12.43	/	11.99	/	11.56	/	11.38	/	10.70	/	10.89	/	9.82	/	12.91	/