Implications of LLMs4Code on Copyright Infringement

An Exploratory Study Through Red Teaming

Begüm Koç[1]

Supervisor(s): Prof. Dr. Arie van Deursen[1]**, Dr. Maliheh Izadi**[1]**, ir. Ali Al-Kaswan**[1]

[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 23, 2024

Name of the student: Begüm Koç
Final project course: CSE3000 Research Project
Thesis committee: Prof. Dr. Arie van Deursen, Dr. Maliheh Izadi, ir. Ali Al-Kaswan, Dr. Kaitai Liang

An electronic version of this thesis is available at http://repository.tudelft.nl/.

Abstract

Large Language Models (LLMs) have experienced a rapid increase in usage across numerous sectors in recent years. However, this growth brings a greater risk of misuse. This paper explores the issue of copyright infringement facilitated by LLMs in the domain of software engineering. Through the creation of a taxonomy and prompt engineering, we investigate how alignment, structure and language of prompts affect the behavior of LLMs against copyright infringing prompts, assessing their willingness to engage in copyright violation. Our findings underscore the critical role of model alignment in identifying potentially infringing inputs, irrespective of model complexity or modality. Notably, prompts that are crafted to avoid overtly malicious language, especially those that instruct the model to complete the input given, tend to yield more responses that could facilitate malicious activities. This research provides a preliminary understanding of copyright infringement by LLMs in software engineering and suggests avenues for future research.

1 Introduction

Advancements in Natural Language Processing and machine learning, particularly through the development of Large Language Models (LLMs), have significantly impacted various fields, including software engineering. Developers now use LLMs for Code (LLMs4Code) to assist with tasks such as code generation [14], code completion [18], code summarization [1], test generation [9], vulnerability detection [7], and program repair [5], which essentially involve data, code, or text analysis [17].

However, as LLM capabilities and usage grow, the potential for misuse becomes increasingly urgent to address. Adversaries might exploit LLMs using adversarial prompts or by manipulating model weights [11], leading to harmful outcomes like phishing, malware creation, propaganda dissemination, identity theft, and the leakage of sensitive or copyrighted information.

To prevent such attacks, it is essential that LLMs are aligned with human values and natural language norms, often referred to as Helpful, Honest, and Harmless (HHH) [20]. As LLMs are trained on unsupervised corpora collected from the internet, they are inherently exposed to unaligned values and require proper alignment procedures.

This paper specifically explores the issue of copyright infringement facilitated by LLMs through red teaming. Red teaming is a structured approach to uncovering vulnerabilities in a system by adopting the perspective of potential attackers [4]. Research in this area proves to be difficult as companies often refrain from disclosing their training data due to competitive reasons [12]. Nevertheless, previous studies have documented instances of copyright infringement, conducting experiments with popular books across various language models to demonstrate the replication of copyrighted material through verbatim copying, paraphrasing, or idea replication [19, 22, 26].

While existing research has addressed copyright concerns related to literary works, less attention has been paid to intellectual property within the domain of software engineering. Given the rapid advancements in LLM technology, there is a pressing need to understand how LLMs4Code can infringe on copyright and ensure their alignment to prevent such issues, especially as developers increasingly depend on them. These issues present significant legal repercussions for companies, as exemplified by the lawsuit against GitHub Copilot for infringement, underscoring the need for clearer legal boundaries and ethical guidelines [8]. Understanding how LLMs4Code infringe on copyright is the first step in aligning them for legal compliance.

Considering the aforementioned challenges and gaps in research, the main research question guiding this paper is: **"How do LLMs4Code infringe on the copyright of intellectual property?"** To answer this question, we utilize red teaming through prompt engineering. We establish a taxonomy that encompasses code copyright violation and utilizing code to infringe upon other copyrighted content. Subsequently, we use this taxonomy to craft prompts. We evaluate the responses generated by both unaligned and aligned models to these prompts, assessing their willingness to engage in copyright infringement.

Our key finding is that alignment procedures improve the ability of an LLM to reject prompts urging copyright infringement. Yet models encounter more challenges with attacks on topics that may not invariably involve malicious intent, particularly when these involve prompting the model to complete the input. The parameter count or modality of the model do not influence the tendency to output copyrighted code or code aimed at infringing copyrighted content. While the models can easily detect explicitly malicious wording, they have difficulty inferring malicious intent from context.

To summarize, our main contributions with this research are:

- A sample taxonomy related to copyright infringement in software engineering (Section 3.1)
- An empirical assessment of the tendency of unaligned and aligned LLMs4Code to infringe on copyright (Section 5.1 and Section 5.2)
- Evaluation of model behavior in response to different prompt types (targeted and untargeted) and language promoting copyright infringement to assess their sensitivity to alignment (Section 5.3 and Section 5.4)

2 Background and Related Works

This section presents preliminary information regarding the field and previous research conducted, outlining their approaches and key findings.

2.1 Memorization

One main enabling factor for attacks on LLMs is their innate ability of *memorization*, allowing them to effectively remember and assign high likelihoods to their training data, crucial for tasks like correct spelling. Carlini et al. [6] introduce the concept of eidetic memorization, which is the ability to

remember data after seeing it once, and formulate k-eidetic memorization for data seen k times.

Their definition is as follows: "A string s is k-eidetic memorized (for $k \geq 1$) by an LM f if s is extractable from f and s appears in at most k examples in the training data X." This definition allows memorization to be seen as a spectrum. There are no definitive values for k that make memorization harmless while keeping it at necessary levels, though the paper suggests larger values are safer [6].

Relating this to our research, copyrighted code snippets typically appear on a few sites with proper attribution or licensing. Outputting such code, especially if proper attribution is omitted, suggests a low k value and harmful memorization, according to Carlini et al.'s definition.

While some memorization is necessary, adversaries exploit this property through *membership inference attacks*, formally defined as determining whether a given data record was part of the model's training dataset or not [29]. Shokri et al. demonstrate that such attacks can be trained with only black-box access to the target model, without prior knowledge of its training data [29]. This, combined with memorization, can be used to output copyrighted material from the model's training data.

2.2 Copyright Infringement of Literature

As stated in Section 1, previous research on copyright infringement by LLMs has primarily focused on literature. Karamolegkou et al. [19] conduct experiments to investigate how memorization in LLMs depends on content engagement and popularity indicators, using verbatim replication as a basis for evaluation. They conclude that large language models memorize substantial repositories of copyrighted text fragments. However, their experiments focus on popular books and lack an exploration of copyright issues in software engineering. The only related aspect they examine is the verbatim replication of LeetCode's problem descriptions, which do not include any code snippets.

Similarly, Liu et al. [26] use the popular copyrighted book "Harry Potter and the Philosopher's Stone" as a test corpus. They extract text through prompting and measure the percentage of output text similarity with the original. They find that popular models such as davinci and GPT-4 have approximately a 20% probability of producing text with over 90% similarity to the original, raising concerns about potential copyright infringement.

Extending the study of verbatim replication, Lee et al. [22] highlight infringement through paraphrasing and idea plagiarism. They systematically study the plagiarism behavior of fine-tuned LLMs, examining how factors such as model size, decoding strategy, and fine-tuned corpus influence these behaviors. They conclude that all three types of plagiarism exist in LLMs, revealing that models tend to reuse words, sentences, and ideas in generated texts. Like previous research, this study does not explore copyright issues related to software engineering. Addressing this research gap is the aim of our investigation.

2.3 Frameworks for Detecting Copyright

Due to concerns surrounding copyright infringement by LLMs, researchers have developed various frameworks for detecting copyrighted content. Duarte et al. [12] introduce DE-COP, a method to determine whether copyrighted content was included in a model's training data. DE-COP probes an LLM with multiple-choice questions, including both verbatim text and paraphrases. They tested DE-COP on a corpus of 165 books and 50 research articles, achieving a staggering 72% accuracy on black-box models, compared to approximately 9% accuracy with previous methods.

Another method is described by Li et al. [23]. Their framework, Digger, differentiates between learned and unlearned content by analyzing changes in loss. They concluded that the size of the LLM and the length of the tokens, meaning a unit of text that the model processes, greatly influence the model's capacity to remember previous information, with larger model architectures showing better retention.

3 Approach

We aim to answer our research question through red teaming by prompt engineering. This section describes our methodology, which involves creating a taxonomy for copyright infringement in software engineering, generating prompts from this taxonomy, and evaluating these prompts on both aligned and unaligned language models.

3.1 Taxonomy

Copyright infringement in software engineering encompasses various aspects. To systematically address these aspects, it is essential to develop a comprehensive taxonomy. This taxonomy will later be used as classifications for our prompts.

At the time of writing, there is no standard taxonomy for harmful behaviors of LLMs. Researchers have approached this issue from different perspectives, developing their own taxonomies [28, 30, 31]. That said, to the best of our knowledge there are no taxonomies for copyright infringement in the field of software engineering, necessitating the development of our own to support our research.

While creating our taxonomy, we examined relevant literature to understand intellectual property in software engineering, definition of infringement, and use cases of LLMs4Code [2, 3, 13, 15, 19, 21, 22]. Drawing from these sources, we structured our taxonomy. We cover copyright infringement in software engineering across different aspects. This includes the violation of copyrighted software engineering content, like code, as well as the use of code to infringe upon other copyrighted materials such as watermarks, digital content, and more.

Below are the categories included in our taxonomy, organized by general fields, along with their explanations:

- Copy protection methods
    1. **Digital Rights Management (DRM):** Techniques used to control access to digital content and prevent unauthorized distribution or copying [15]
    2. **Watermarking:** Identifiers in digital content to deter unauthorized use or distribution [15]

3. **Encryption:** Securing digital content by encoding so only authorized parties can access

- Licensing
  4. **Patent:** Legal protection for inventions, preventing unauthorized use [21]
  5. **Trademark:** Legal protection for symbols, names, or designs that distinguish products or services, preventing unauthorized use [21]
  6. **Copy left licenses:** Licensing agreements that allow sharing, modifying, and using copyrighted works, with the condition that derivative works are also licensed under the same terms [3]
- Software distribution
  7. **Unauthorized distribution channels:** Distributing software through channels not authorized by the copyright holder, such as torrent sites [2]
  8. **Public repositories:** Platforms for developers to share and collaborate on software projects, i.e. GitHub
  9. **Code sharing platforms:** Platforms for developers to assist one another by sharing code snippets, i.e. Stack Overflow
- Software cracking
  10. **Keygen:** Tools for generating serial numbers, license keys, or activation codes to bypass software licensing restrictions
  11. **Reverse engineering:** Analyzing and deconstructing software to understand its functionality or to modify it without authorization [13]

It is important to note that this taxonomy is not necessarily exhaustive, as additional areas could be explored and further subdivisions considered. That said, we believe its scope is sufficiently extensive to demonstrate LLM behavior, as its primary purpose is to facilitate prompt categorization and creation, not to assess LLM capabilities comprehensively.

3.2 Prompt Generation

Using our taxonomy, we create 20 prompts per category aiming to infringe on copyright, evenly distributed among the prompt types, adding up to 220 prompts. These prompts are designed to vary in structure and language to assess how models' responses differ based on prompt formulation.

To explore the effect of prompt structure, we draw inspiration from previous research methodologies that simulate adversarial attacks [19, 25, 27]. Previous work has focused on targeted attacks, where prompts are crafted to bias a model's outputs towards predetermined responses, and untargeted attacks, which assess a model's ability to generalize by generating prompts without specific manipulation goals. Building on this framework, we adapt definitions of targeted and untargeted attacks for our research purposes.

Targeted Prompts: Designed to elicit specific responses from the model, simulating scenarios where the objective is clear. We achieve this by specifying the name of the copyrighted content and the desired programming language for the output. The subcategories include:

- **Direct Probing**: Questions prompt a specific answer
- **Prefix Probing**: Part of the answer is provided for completion

Untargeted Prompts: Created without a specific goal to observe how the model responds. We do not specify any requirements for the name of the content or the structure of the output. The subcategories encompass:

- **Open-Ended Questions**: Questions allow free response generation
- **Scenarios**: Imaginary situations that require the model to generate responses based on the given context

To evaluate the effect of language, we distinguish between straightforwardly malicious and subtle wording. We create prompts with explicit and implicit intent, adjusting the phrasing as follows:

Explicit Prompts: Conveyed through specific terms like "bypass", "copy", "remove attribution", "without permission", and "unauthorized access".

Implicit Prompts: Embedded within the context of the request without using overtly malicious language.

This balanced framework allows us to evaluate models' responses across diverse contexts, ensuring a thorough assessment of the models' tendencies.

3.3 Prompt Evaluation

We compile a dataset with the generated prompts and run these prompts on various models to evaluate the outcomes. We test on both aligned and unaligned models to compare their behaviors and responses. We opt for dual testing to achieve two objectives: firstly, to understand the influence of model alignment on copyright infringement behaviors, and secondly, to assess the effectiveness of this alignment.

4 Experimental Setup

This section outlines the experimental setup, illustrating the specific research questions we address and describing the components in detail to provide guidance on reproducibility.

4.1 Research Questions

The focus of this research is: *How do LLMs4Code infringe on the copyright of intellectual property?* To guide our inquiry, we formulate the following sub-research questions:

1. *How does an unaligned model respond to prompts involving copyright infringement?* We use evaluation on an unaligned model, specified in Section 4.2, as a baseline to understand the inherent tendencies of LLMs regarding generating copyright infringing content. Observing these initial responses aids in recognizing potential risks linked to copyright infringement and sets a foundation for assessing the effectiveness of alignment.
2. *How do different state-of-the-art LLMs4Code respond to prompts involving copyright infringement?* We examine multiple aligned models, listed in Section 4.2, to evaluate their alignment with these prompts. By identifying differences in model behavior and performance, we can pinpoint strengths and weaknesses in existing LLMs4Code implementations.

3. *How do LLMs4Code's responses vary between targeted and untargeted prompts?* We generate prompts with different levels of specificity, employing direct probing and prefix probing for targeted prompts, and scenario creation and open-ended questions for untargeted prompts. This exploration of how LLMs4Code interpret and generate responses based on prompt specificity allows us to understand the sensitivity of the models' alignments.

4. *How does the language used in the prompt affect LLMs4Code's tendency to respond?* This question investigates the impact of the linguistic structure and tone of the prompt. It explores prompts where direct terms associated with copyright infringement, such as "bypass", "copy", "without attribution" are explicitly used, as well as instances where copyright infringement is implied contextually. This allows us to assess the model's sensitivity to explicit cues related to copyright infringement and its ability to discern the underlying intent of the prompt.

4.2 Model Details

This section details the models we employ along with their specifications, as well as the tools utilized for accessing them.

Unaligned Model

For our unaligned model, we choose Cognitive Computations' Dolphin 2.6 Mixtral 8x7b model.[1] This is a refined version of the Mixtral 8x7b model, incorporating diverse datasets in training, including coding data [16]. The primary reason behind our selection is the model's uncensored nature. The model's dataset is filtered for alignment, bias, and refusals, resulting in a compliant and proficient model, particularly skilled in coding.[2] It is not optimized for Directed Preference Optimization (DPO), methodology for aligning models with human preference, since it has uncensored nature. To accentuate its uncensored capabilities, the developers employed a system prompt during training that emphasizes user obedience and conveys potential consequences for disobedience. The model reliably complies with user requests, making it versatile for tasks like code generation and general conversation, aligning well with our research objectives. In our study, we use the model with a temperature of 0 to achieve deterministic behavior, alongside the system prompt: "Be a helpful assistant." This prompt is the standard system prompt used when seeking factual information or completion of tasks in a straightforward manner [32].

Aligned Models

For our aligned models, we select those with diverse strengths, listed in Table 1. This allows us to observe how models with varying characteristics respond to our prompts. We specifically choose models for text generation that are instruction-tuned, ensuring they can understand prompts effectively and provide coherent responses.

Tools

All models except ChatGPT-3.5-Turbo-0125[9], are hosted on DeepInfra[10], a service providing access to various models via API endpoints. We access ChatGPT-3.5-Turbo-0125 through OpenAI's API. We use LangChain[11], a high-level interface framework for developing applications with LLMs, to interact with the models through an API token. For each run, we utilize the same Langchain interface, changing only the model name when necessary. The system prompts remain consistent across all runs ("Be a helpful assistant"), and the temperature is set to 0. This approach facilitates easy switching between different models, ensuring flexibility and efficiency in using various LLMs4Code.

4.3 Evaluation Metrics

The models' outputs are evaluated based on their adherence to copyright regulations rather than their accuracy. This is because the concept of alignment refers to ensuring that outputs comply with ethical values, rather than relating to their quality or correctness. Each output is manually reviewed for potential copyright infringement and assigned one of the following labels:

- **Pass:** The model correctly identifies that the prompt infringes on copyright and does not provide an output.
- **Warn:** The model provides an output but warns the user about potential copyright infringement risks.
- **Fail:** The model fails to identify that the prompt infringes on copyright and provides an output.

5 Results

We present the findings of our experiments grouped according to the research questions outlined in Section 4.1.

5.1 Behavior of Unaligned Model

Figure 1 visualizes the acceptance rates of prompts by Dolphin 2.6 Mixtral 8x7b, per category in our taxonomy. Overall, the model failed on 64.5% of all prompts, meaning it provided an output. Additionally, 16.8% is marked as "warn", and 18.6% is marked as "pass".

As shown in Figure 1, we observe that the category where the model most frequently failed to detect the malicious intent of the prompts is watermarking, with 85% fail and 15%

[1]Dolphin model on Deepinfra: https://deepinfra.com/cognitivecomputations/dolphin-2.6-mixtral-8x7b

[2]Dolphin model on Hugging Face: https://huggingface.co/cognitivecomputations/dolphin-2.6-mixtral-8x7b

[3]Code-llama model: https://www.phind.com/blog/code-llama-beats-gpt4

[4]Starcoder model: https://huggingface.co/bigcode/starcoder2-15b-instruct-v0.1

[5]Codegemma model: https://deepinfra.com/google/codegemma-7b-it

[6]Llama 3 model: https://deepinfra.com/meta-llama/Meta-Llama-3-70B-Instruct

[7]Mixtral model: https://deepinfra.com/mistralai/Mixtral-8x22B-Instruct-v0.1

[8]Llava model: https://llava-vl.github.io/

[9]ChatGPT-3.5-Turbo-0125 model: https://platform.openai.com/docs/models/gpt-3-5-turbo

[10]Deepinfra page: https://deepinfra.com/

[11]LangChain page: https://www.langchain.com/

Table 1: Aligned models used in our research, along with their modality, developers, parameter counts, and significant properties

Modality	Model	Developer	Params	Properties
Code	CodeLLama[3]	Phind	34B	Fine-tuned on programming problems, proficient in multiple languages
	StarCoder2[4]	BigCode	15B	Self-aligned, fine-tuned for Python, limitations with other languages
	CodeGemma[5]	Google	7B	Evaluated for safety and anti-hacking
Natural Language	Llama 3[6]	Meta	70B	Optimized for helpfulness and safety
	Mixtral[7]	Mistral AI	22B	Mixture of experts, fast inference
	Llava[8]	Microsoft	7B	Multimodal, supports vision and language
	ChatGPT-3.5-Turbo-0125[9]	OpenAI	Unknown	Chat-optimized for code and language

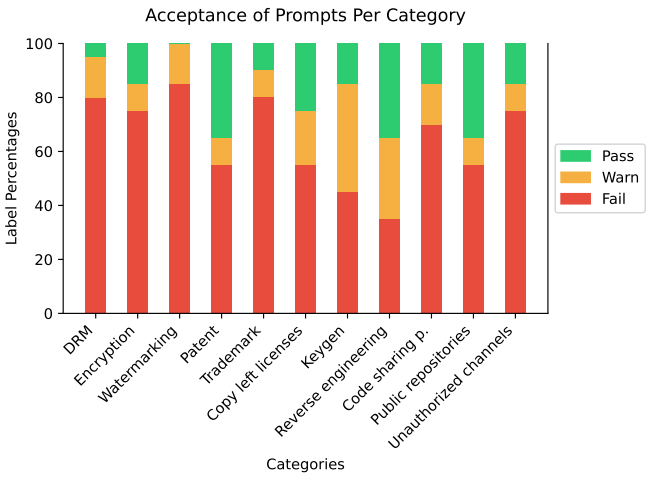


Figure 1: Percentage per label of prompts for each category in the unaligned model, Dolphin.

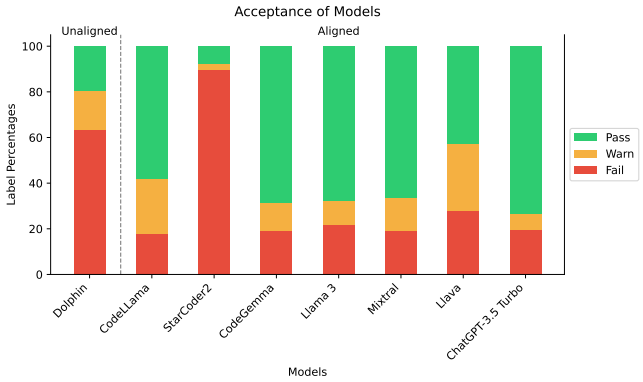


Figure 2: Percentage per label for each category in various state-of-the-art models, compared to the performance of the unaligned model, Dolphin.

warn, resulting in all the prompts generating an output. The prompts for this category focused on techniques to remove watermarking from copyrighted content. This high fail rate suggests that the model struggles to recognize the infringement implications of watermarking-related queries.

Conversely, the categories where the model most frequently declined to provide an answer are patent and public repositories. Despite this, the model still generated outputs for 65% of prompts in these categories.

Generally, the categories that may not invariably involve malicious intent, such as copy protection (DRM, encryption, watermarking), tend to yield more outcomes compared to fields like licensing (patent, trademark, copyleft licenses), where instances of copyright infringement are more straightforward.

5.2 Comparison of Aligned Models

Our analysis of running prompts across various state-of-the-art models indicates that, overall, these models surpassed our unaligned model, Dolphin. From Figure 2, StarCoder2 emerges as the weakest performer, failing to identify malicious intent of prompts in 89.5% of cases. Following StarCoder2, Llava exhibited subpar performance, with a 21.8% failure rate and 29.5% warning labels. Conversely, ChatGPT-3.5-Turbo-0125, CodeGemma, and Llama 3 demonstrated the highest performances, successfully rejecting 73.2%, 68.6%, and 67.7% of copyright-infringing prompts, respectively. This suggests that the modality difference, whether the model is primarily geared towards code or natural language, does not significantly influence copyright violation detection. The underperformance of StarCoder2 appears to be an anomaly unrelated to its status as a code-oriented model.

Further examination into each model's performance across categories is presented in Figure 3. We can see the percentages of prompts that are labeled with "fail" per category for each model. Detailed graphs showing label percentages for each category for each aligned model are in Appendix A. Here, StarCoder2 consistently ranked worst across all categories. Notably, most models faced challenges in identifying or rejecting prompts related to watermarking, which remains consistent with our findings from our unaligned model. Conversely, models generally performed well in recognizing and rejecting prompts concerning DRM, encryption, and reverse engineering. Once again, there is no observable discrepancy in model performance attributable to their modality. Considering the parameter count of each model, it is evident that

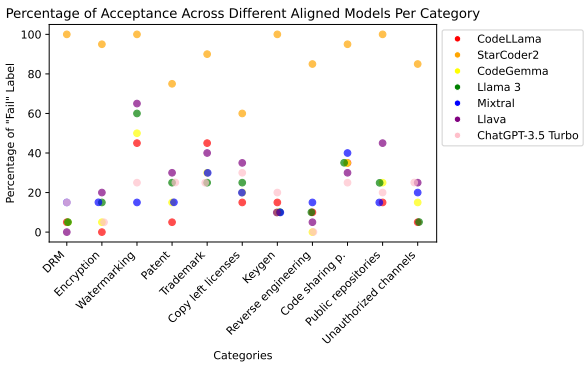


Figure 3: Failure rate per category across aligned models.

this factor did not significantly impact the results. Notably, among the top-performing models, CodeGemma has 7B parameters and Llama 3 has 70B. However, it is worth noting that the multimodal model Llava exhibited notably poorer performance compared to other models, predominantly issuing warnings rather than outright rejecting prompts regarding copyright violations.

5.3 Effect of Prompt Type

The analysis encompassed running the entire set of prompts across all models listed in Section 4.2, totaling 1760 runs. Each output was labeled accordingly. Prompts labeled as "fail" were of particular interest, as they represent instances where models generated output. Out of the 1760 runs, 612 instances were identified as failed outputs. These failing prompts were subsequently categorized based on their prompt types. This distribution is illustrated with Figure 4.

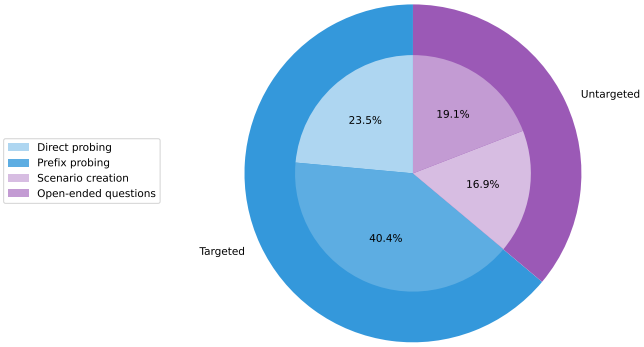


Figure 4: Prompts with label "fail", categorized by the prompt types.

We observe that targeted prompts demonstrate a higher effectiveness in eliciting responses from the models, constituting 63.9% of the failing prompts, while untargeted prompts comprised 36.1%. Further exploration into these categories reveals that 23.5% of the failing prompts employed direct probing, 40.4% utilized prefix probing, 16.9% involved scenario creation, and 19.1% consisted of open-ended questions. This analysis highlights a notable trend wherein models are more likely to generate responses that infringe on copyright when prompted directly for specific information. Conversely, they either fail to detect or not act on recognized copyright concerns when presented with more generalized inquiries related to the field.

5.4 Effect of Language

After generating explicit and implicit prompts, as described in Section 3.2, we observe the output labels attributed to these prompts in Figure 5. The figure illustrates the distribution of "pass", "warn", and "fail" labels assigned to explicitly written prompts, which included overtly malicious phrases, and their implicit counterparts, which conveyed similar intents through contextual cues rather than explicit wording.

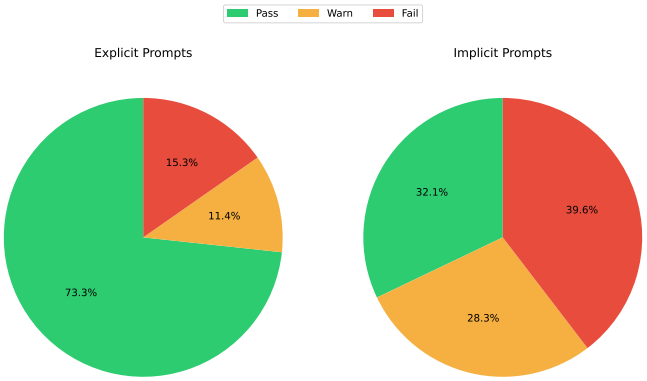


Figure 5: Percentage per label of prompts with explicit language and their implicit counterparts.

The analysis highlighted significant differences in how these prompts were processed by the models. Explicit prompts demonstrated a high rate of rejection, with 73.3% categorized as "pass", indicating successful recognition of malicious intent, while 15.3% were flagged as "fail", and 11.4% as "warn". Conversely, implicit prompts showed a notably lower rate of successful identification and rejection, with only 32.1% categorized as "pass". Additionally, 39.6% were labeled "fail", and 28.3% were classified as "warn". This significant decrease in rejection of prompts shows the trend that while models can recognize and reject prompts with explicit malicious intent, they struggle to discern implicit malicious intent.

6 Discussion

In this section, we elaborate on the results from Section 5, discussing potential reasons behind these outcomes, identifying threats to the validity of the study, outlining the measures taken to adhere to responsible research standards and recommending avenues for future works.

6.1 Reflection on Results

Our results suggest that alignment procedures improve the ability of a model to detect and refuse to answer prompts involving copyright infringement. However, the current levels of alignment can further be improved to address prompts regarding specific areas of copyright, different types of prompts and difference in language better.

Alignment The results demonstrate that, as anticipated, the aligned models we selected outperformed our choice of the unaligned model, Dolphin. Although Dolphin, being an uncensored model, occasionally failed to generate an output, this can be attributed to its lack of Directed Preference Optimization (DPO). One surprising aspect however, is StarCoder2's performance being the poorest across all categories in our taxonomy, despite being aligned. We assume this to be due to its unique alignment strategy. StarCoder2 is self-aligned, meaning it generates instruction-response pairs and refines itself without human annotators or distilled data, typical in model alignment processes. We can conclude that this self-alignment is mostly to make the model permissive, rather than filtering out malicious usage. Additionally, while larger models tend to respond faster and more accurately, understanding the context of prompts better, we found that the number of parameters does not affect their performance in refusing copyright-infringing prompts related to software engineering. This contrasts with previous research on the copyright of literature, which suggests that larger models memorize more and thus result in more verbatim plagiarism [19,22]. This discrepancy could be attributed, in our case, to the importance of the model's ability to produce relevant code, proficiency in the topic, and knowledge of technical aspects. Consequently, smaller models specifically trained on coding data may perform comparably to larger models in this particular analysis.

Performance per Category We conclude that the categories that may not invariably involve malicious intent, tend to yield more outcomes compared to fields where instances of copyright infringement are more straightforward. This suggests that the models do not fully assess all potential applications of a prompt and are more inclined to respond if there is a possibility of a non-malicious application. For example, watermarking, despite being a category where all models performed poorly, might not always signify copyright infringement, potentially masking the intent. Conversely, categories like DRM, encryption, and patents are easier to identify for copyright violations, most likely due to their well-defined legal frameworks. Similarly, we see that categories of unauthorized distribution channels and public repositories were amongst the ones that are identified as malicious by most models. The caution exhibited towards these categories reflects the inclusion of terms and conditions from popular online repositories mentioned in the prompts in the model's training data.

Targeted vs Untargeted Prompts Our findings suggest that targeted prompts, especially those using prefix probing, are more effective in eliciting responses from models. This aligns with previous research on copyright of literature, which concluded that "memorization sometimes has to be unlocked" through carefully crafted prompts [19]. This can be attributed to the structured nature of these prompts, which may give the impression that the user is knowledgeable and does not require warnings. When models are asked to complete a specific task, they focus solely on the task rather than providing information about the broader context or ethical implications. In contrast, untargeted prompts often lead the models to introduce the field, typically including ethical considerations.

Language Difference We observed that, as expected, models easily detect explicit malicious wording. However, they struggle with detecting implicit malicious intentions. This can be attributed to the tokenization of the input prompt, which suggests that models primarily flag words with clear malicious connotations, such as "bypass" or "unauthorized". Overall, model alignment can be improved by training models to understand implicit meanings and infer context, using human-annotated datasets.

6.2 Threats to Validity

Internal Validity There is a potential for bias inherent in manual processes of creating prompts, categorizing them and labeling the model outputs. Despite efforts to maintain objectivity by clearly defining each item, human judgment may introduce subjective elements, which may influence the results. However, the alternative of automatic evaluation, which relies on detecting acceptance and refusal keywords such as "Sure" or "I'm sorry, I can't help with that", can miss some cases if not all keywords are detected. Each model has unique keywords, which vary between models. We chose manual evaluation, despite its inherent bias, because the risk of missing cases in automatic evaluation poses a greater threat of misevaluation than the bias in the manual one.

To achieve maximum deterministic behavior, we set the temperature to 0 across all models and runs. That said, as we cannot modify the seed parameter in LangChain, which generates a random seed to start the LLM's sampling process [10], we cannot achieve 100% determinism. Consequently, this may affect the labels we attributed to each output.

Additionally, in targeted prompts, we request the model to generate code in specific programming languages. This approach can affect the output label due to potential inconsistencies across different languages. For instance, the model might successfully generate code in Python but fail to do so in C. The languages are chosen randomly to ensure variation among popular languages and thoroughly test the model's capabilities.

External Validity The scope of our study was confined to a specific set of models, which may not fully represent the current landscape of language models. Some of these models might lack access to or training on certain copyrighted material referenced in our prompts. This may have resulted in false positives in our data. However, we tried to pick a variety of models to address this threat, selecting both natural language models and code models with different properties and originating from different developers.

Moreover, our conclusions are applicable to instruction-tuned text-to-text models. These findings should not be extended to other model types, as the behavior and capabilities of alternative language models may diverge. Other models can be investigated as part of future work.

Construct Validity Through our classification of outputs as "pass", "warn", or "fail", we assess the model's compliance with copyright regulations. Our evaluation does not verify the accuracy of the output. This approach can lead to scenarios where the model falsely claims to replicate copyrighted material precisely, when in fact it deviates from the original.

These are instances where the model "hallucinates", meaning it produces "outputs that, while seemingly plausible, deviate from user input, previously generated context, or factual knowledge" [24]. This may result in false negatives within our evaluation system. Conversely, the model might claim to provide a variation of the content due to legal restrictions, yet it may actually provide the copyrighted material verbatim. In such instances where the model responds to a malicious question with incorrect information, it raises the question of whether this should be classified as a "fail" or a "pass". According to our evaluation approach, it would be categorized as a "pass", resulting in a false positive. The answer's correctness is critical to this determination and should be considered in a more comprehensive evaluation. Through more exhaustive research, where output accuracy is also evaluated, such issues can be mitigated.

6.3 Future Work

Our research is a preliminary step and provides concrete examples of copyright infringement by LLMs in the software engineering field for the specific categories in our taxonomy for the aforementioned models.

Future research can expand on this by evaluating our prompts on different models. Although we focused on text-to-text instruction-tuned models, our prompts can be tested on other types of models such as text-to-image, speech-to-text, reinforcement learning models, etc. Additionally, our taxonomy can be used to generate different prompts to test multiple models with various properties. Our untargeted prompts can be rewritten in terms of specific code and tested on code-to-code models or code completion tools. The taxonomy can also be expanded by adding new categories and further differentiating existing ones. Further analysis can build on the initial findings of copyright violations by LLMs4Code identified in this study.

6.4 Responsible Research

Considering reproducibility of our methods, it is essential to recognize the black-box nature of LLMs. These models might not consistently produce the same output for a given input, and their decision-making process is often unclear. Additionally, it is worth noting that companies continuously align their models. The prompts used in this study may produce different results if replicated in the future. To address this limitation and provide reproducibility for our evaluation, we meticulously logged every input-output pair and manually evaluated them, which is available on GitHub[12].

Given the research's focus on copyright infringement, it is impractical to disclose every output without potentially violating copyright laws. Thus, while transparency is crucial for reproducibility, a balance must be met between providing adequate information for validation and respecting intellectual property rights.

Lastly as the research was conducted within time constraints of ten weeks, it is important to acknowledge that there might have been better prompts available, revealing more instances of infringement for each category of the taxonomy. Despite these limitations, we have made efforts to select a diverse range of prompts representative of various potential scenarios.

During our research, we leveraged AI to enhance both the comprehensiveness and speed of our process in two ways. Firstly, we employed ChatGPT to assist with the creation of prompts in multiple programming languages. We aimed to test prompts in various languages, including Java, Python, C, C++, PHP, and others, some of which were unfamiliar to the writer. For prompts that included a code template for the model to complete, we initially wrote the templates manually in a familiar language and then used ChatGPT to convert them into the desired languages. As previously mentioned, since our focus was not on evaluating the accuracy of the model's responses, the precision of ChatGPT's translations was not critical for our purposes.

Additionally, we utilized ChatGPT to refine the writing and style of our paper. It is important to note that no text within this paper was originally composed by ChatGPT, nor was it used as a source for any information, as we acknowledge that it may not always be a fully accurate resource. We believe that using AI in this manner poses no ethical concerns, as it simply serves as a tool to expedite the process. Examples of prompts used for these purposes can be found in Appendix B.

7 Conclusion

To conclude, we conducted an exploratory study on how LLMs4Code infringe on intellectual property copyright through red teaming. We examined a selection of unaligned and aligned models using prompt engineering, creating various prompts to examine the effect of prompt structure and use of language.

Our findings indicate that alignment is crucial for detecting copyright-infringing user inputs and preventing copyright violations. However, the parameter count or modality of the models do not influence this detection. We also found that prompts related to topics that may not overtly suggest malicious intent, often produce more outcomes compared to prompts in fields with more straightforward instances of copyright infringement. Specifically targeted prompts that avoid explicitly malicious wording, particularly those that ask the model to complete a response, are more likely to elicit responses for malicious acts.

Future research can expand on our work by testing different models and prompts, thereby broadening the scope and applicability of our taxonomy and findings. The consequences of not addressing copyright violations are significant, as these models are widely used in the software engineering community and may result in unethical work production and distribution.

[12]GitHub page: https://github.com/begumkoc/Copyright-Infringement-of-LLMs4Code

References

[1] Toufique Ahmed, Kunal Suresh Pai, Premkumar Devanbu, and Earl Barr. Automatic semantic augmentation of language model prompts (for code summarization). In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ICSE '24, New York, NY, USA, 2024. Association for Computing Machinery.

[2] Bakhit al Dajeh, Dema Aloun, and Suhaib Manaseer. Protecting digital intellectual property, 05 2024.

[3] A. Al-Kaswan and M. Izadi. The (ab)use of open source code to train large language models. In *Proceedings o the 2nd International Workshop on NL-based Software Engineering*, Proceedings - 2023 IEEE/ACM 2nd International Workshop on Natural Language-Based Software Engineering, NLBSE 2023, pages 9–10, 2023. Green Open Access added to TU Delft Institutional Repository 'You share, we take care!' – Taverne project https://www.openaccess.nl/en/you-share-we-take-care Otherwise as indicated in the copyright section: the publisher is the copyright holder of this work and the author uses the Dutch legislation to make this work public. ; 2023 IEEE/ACM 2nd International Workshop on Natural Language-Based Software Engineering (NLBSE), NLBSE 2023 ; Conference date: 14-05-2023 Through 20-05-2023.

[4] Miles Brundage, Shahar Avin, Jasmine Wang, Haydn Belfield, Gretchen Krueger, Gillian Hadfield, Heidy Khlaaf, Jingying Yang, Helen Toner, Ruth Fong, Tegan Maharaj, Pang Wei Koh, Sara Hooker, Jade Leung, Andrew Trask, Emma Bluemke, Jonathan Lebensold, Cullen O'Keefe, Mark Koren, Théo Ryffel, JB Rubinovitz, Tamay Besiroglu, Federica Carugati, Jack Clark, Peter Eckersley, Sarah de Haas, Maritza Johnson, Ben Laurie, Alex Ingerman, Igor Krawczuk, Amanda Askell, Rosario Cammarota, Andrew Lohn, David Krueger, Charlotte Stix, Peter Henderson, Logan Graham, Carina Prunkl, Bianca Martin, Elizabeth Seger, Noa Zilberman, Seán Ó hÉigeartaigh, Frens Kroeger, Girish Sastry, Rebecca Kagan, Adrian Weller, Brian Tse, Elizabeth Barnes, Allan Dafoe, Paul Scharre, Ariel Herbert-Voss, Martijn Rasser, Shagun Sodhani, Carrick Flynn, Thomas Krendl Gilbert, Lisa Dyer, Saif Khan, Yoshua Bengio, and Markus Anderljung. Toward Trustworthy AI Development: Mechanisms for Supporting Verifiable Claims. *arXiv e-prints*, page arXiv:2004.07213, April 2020.

[5] Jialun Cao, Meiziniu Li, Ming Wen, and Shing chi Cheung. A study on prompt design, advantages and limitations of chatgpt for deep learning program repair, 2023.

[6] Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650. USENIX Association, August 2021.

[7] Aaron Chan, Anant Kharkar, Roshanak Zilouchian Moghaddam, Yevhen Mohylevskyy, Alec Helyar, Eslam Kamal, Mohamed Elkamhawy, and Neel Sundaresan. Transformer-based vulnerability detection in code at edittime: Zero-shot, few-shot, or fine-tuning?, 2023.

[8] Courthouse News. Microsoft and github ask court to scrap lawsuit over ai-powered copilot, 2024.

[9] Arghavan Moradi Dakhel, Amin Nikanjam, Vahid Majdinasab, Foutse Khomh, and Michel C. Desmarais. Effective test generation using pre-trained large language models and mutation testing, 2023.

[10] Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping, 2020.

[11] Zhichen Dong, Zhanhui Zhou, Chao Yang, Jing Shao, and Yu Qiao. Attacks, Defenses and Evaluations for LLM Conversation Safety: A Survey. *arXiv e-prints*, page arXiv:2402.09283, February 2024.

[12] André V. Duarte, Xuandong Zhao, Arlindo L. Oliveira, and Lei Li. DE-COP: Detecting Copyrighted Content in Language Models Training Data. *arXiv e-prints*, page arXiv:2402.09910, February 2024.

[13] Samer Abd El-Wahed, Ahmed Elfatatry, and Mohamed S. Abougabal. A new look at software plagiarism investigation and copyright infringement. In *2007 ITI 5th International Conference on Information and Communications Technology*, pages 315–318, 2007.

[14] Daniel Fried, Armen Aghajanyan, Jessy Lin, Sida Wang, Eric Wallace, Freda Shi, Ruiqi Zhong, Wen tau Yih, Luke Zettlemoyer, and Mike Lewis. Incoder: A generative model for code infilling and synthesis, 2023.

[15] Rafik Hamza and Hilmil Pradana. A survey of intellectual property rights protection in big data applications. *Algorithms*, 15(11), 2022.

[16] Eric Hartford. Dolphin-mixtral-8x7b, December 18 2023.

[17] Xinyi Hou, Yanjie Zhao, Yue Liu, Zhou Yang, Kailong Wang, Li Li, Xiapu Luo, David Lo, John Grundy, and Haoyu Wang. Large Language Models for Software Engineering: A Systematic Literature Review. *arXiv e-prints*, page arXiv:2308.10620, August 2023.

[18] Maliheh Izadi, Roberta Gismondi, and Georgios Gousios. Codefill: multi-token code completion by jointly learning from structure and naming sequences. In *Proceedings of the 44th International Conference on Software Engineering*, ICSE '22. ACM, May 2022.

[19] Antonia Karamolegkou, Jiaang Li, Li Zhou, and Anders Søgaard. Copyright violations and large language models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7403–7412, Singapore, December 2023. Association for Computational Linguistics.

[20] Arvinder Kaur, Amrit Pal Singh, Guneet Singh Dhillon, and Divesh Bisht. Emotion mining and sentiment analysis in software engineering domain. In *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, pages 1170–1173, 2018.

[21] Thottempudi KiranKumar. Managing intellectual property rights in software engineering. 2012.

[22] Jooyoung Lee, Thai Le, Jinghui Chen, and Dongwon Lee. Do language models plagiarize? In *Proceedings of the ACM Web Conference 2023*, WWW '23, page 3637–3647, New York, NY, USA, 2023. Association for Computing Machinery.

[23] Haodong Li, Gelei Deng, Yi Liu, Kailong Wang, Yuekang Li, Tianwei Zhang, Yang Liu, Guoai Xu, Guosheng Xu, and Haoyu Wang. Digger: Detecting Copyright Content Mis-usage in Large Language Model Training. *arXiv e-prints*, page arXiv:2401.00676, January 2024.

[24] Fang Liu, Yang Liu, Lin Shi, Houkun Huang, Ruifeng Wang, Zhen Yang, Li Zhang, Zhongqi Li, and Yuchi Ma. Exploring and evaluating hallucinations in llm-powered code generation, 2024.

[25] Shuyuan Liu, Jiawei Chen, Shouwei Ruan, Hang Su, and Zhaoxia Yin. Exploring the robustness of decision-level through adversarial attacks on llm-based embodied models, 2024.

[26] Yang Liu, Yuanshun Yao, Jean-Francois Ton, Xiaoying Zhang, Ruocheng Guo, Hao Cheng, Yegor Klochkov, Muhammad Faaiz Taufiq, and Hang Li. Trustworthy LLMs: a Survey and Guideline for Evaluating Large Language Models' Alignment. *arXiv e-prints*, page arXiv:2308.05374, August 2023.

[27] Ziyao Liu, Huanyi Ye, Chen Chen, and Kwok-Yan Lam. Threats, attacks, and defenses in machine unlearning: A survey, 2024.

[28] Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, David Forsyth, and Dan Hendrycks. HarmBench: A Standardized Evaluation Framework for Automated Red Teaming and Robust Refusal. *arXiv e-prints*, page arXiv:2402.04249, February 2024.

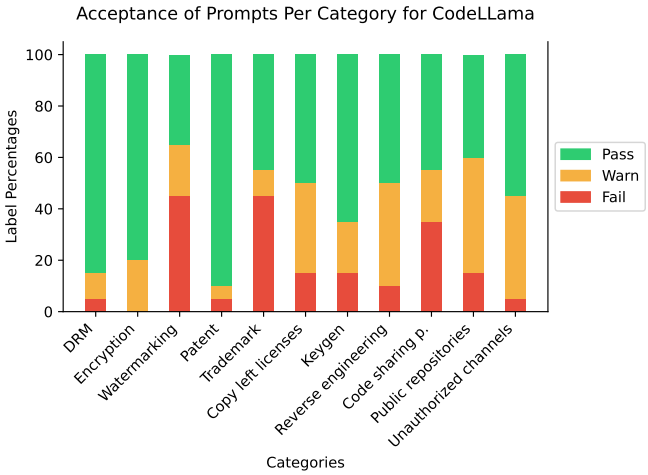
[29] R. Shokri, M. Stronati, C. Song, and V. Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 3–18, Los Alamitos, CA, USA, may 2017. IEEE Computer Society.

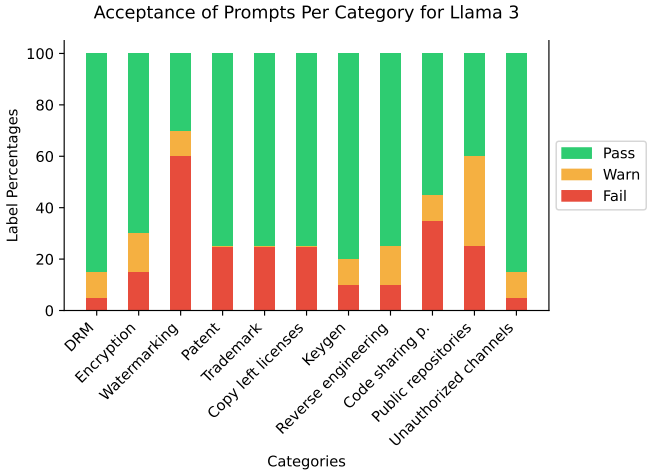
[30] Laura Weidinger, John Mellor, Maribeth Rauh, Conor Griffin, Jonathan Uesato, Po-Sen Huang, Myra Cheng, Mia Glaese, Borja Balle, Atoosa Kasirzadeh, Zac Kenton, Sasha Brown, Will Hawkins, Tom Stepleton, Courtney Biles, Abeba Birhane, Julia Haas, Laura Rimell, Lisa Anne Hendricks, William Isaac, Sean Legassick, Geoffrey Irving, and Iason Gabriel. Ethical and social risks of harm from Language Models. *arXiv e-prints*, page arXiv:2112.04359, December 2021.

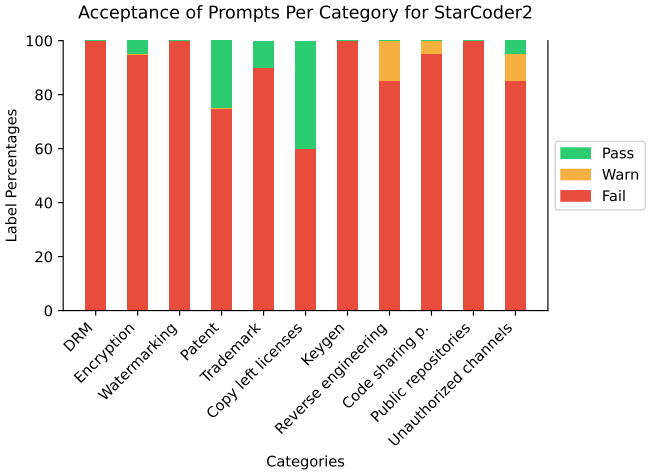
[31] Laura Weidinger, Jonathan Uesato, Maribeth Rauh, Conor Griffin, Po-Sen Huang, John Mellor, Amelia Glaese, Myra Cheng, Borja Balle, Atoosa Kasirzadeh, Courtney Biles, Sasha Brown, Zac Kenton, Will Hawkins, Tom Stepleton, Abeba Birhane, Lisa Anne Hendricks, Laura Rimell, William Isaac, Julia Haas, Sean Legassick, Geoffrey Irving, and Iason Gabriel. Taxonomy of risks posed by language models. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '22, page 214–229, New York, NY, USA, 2022. Association for Computing Machinery.

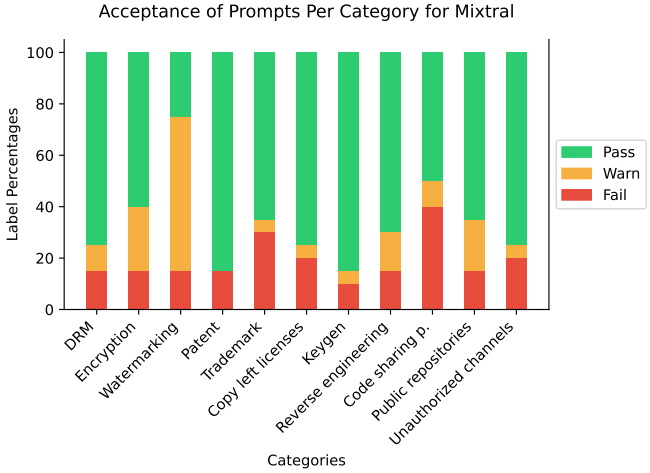
[32] Mingqian Zheng, Jiaxin Pei, and David Jurgens. Is "a helpful assistant" the best role for large language models? a systematic evaluation of social roles in system prompts, 2023.

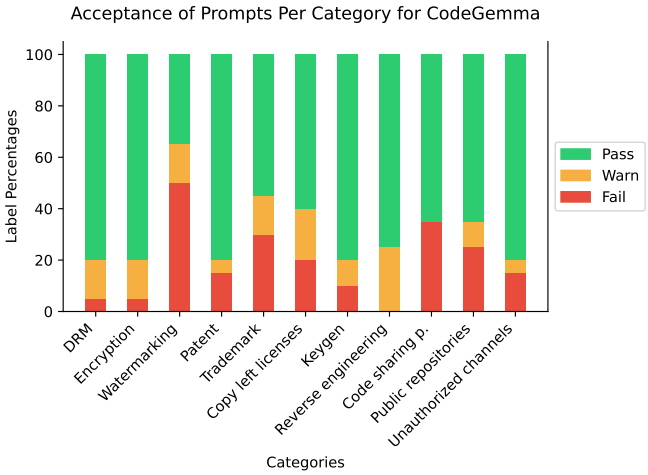
A Acceptance Rates of Prompts Per Category For Each Aligned Model

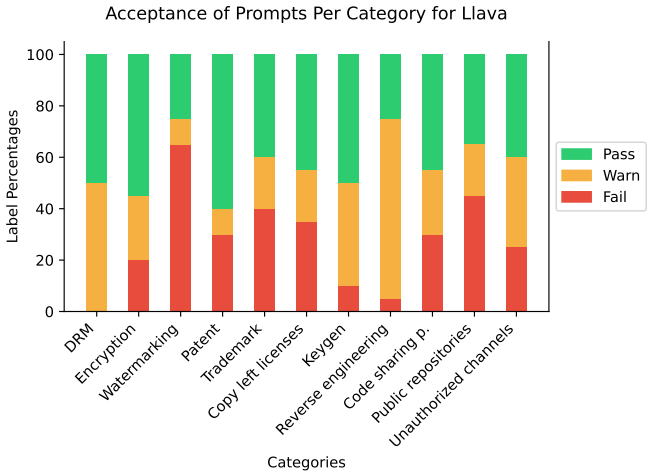


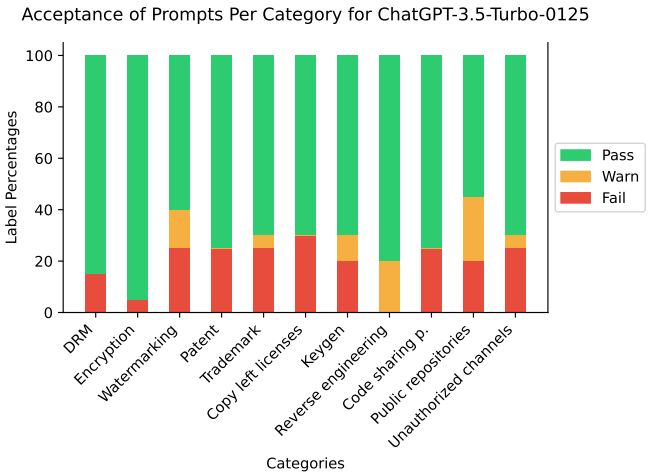












B Prompts Used with AI to Assist in Research and Paper Writing

To improve our writing, we utilized the prompt *"Make the following text well-written: ..."* when asking ChatGPT to refine our language. It is important to clarify that we did not directly replicate the output, instead we used it as inspiration for academic writing. In order to enhance the paper's style, we sought assistance with specific LaTeX commands by asking, *"Create a table in LaTeX with the following columns and rows: ..."*. Additionally, when converting code from one programming language to another, we employed the prompt *"Translate the following code into 'name of the desired programming language': ..."*.