Utrecht University

Enhancing the ECSER pipeline: Evaluating Large Language Model Classifiers in SE Research

Ruben van der Zeijden (7081111)

Supervisor: Dr. F.B. Aydemir
Second Reader: Dr. D. Dell'Anna

Abstract

The various research papers in the field of Software Engineering (SE) that use classification algorithms, LLMs, or other machine learning methods to obtain their results differ in how many and which evaluation metrics are reported, whether or not significance tests are performed, and which steps are taken to aid reproducibility. The ECSER pipeline was designed to mitigate this issue by providing a step-by-step pipeline that researchers can use to report their results when classification algorithms are used, and the pipeline was empirically shown to be effective in replicating the findings of several studies, as well as producing additional findings and occasionally contradicting the findings of the original papers. However, the ECSER pipeline is designed for evaluating simple classifiers and gives no specific recommendations for LLM classifiers, despite LLMs being an increasingly popular choice for classification tasks. The goal of this thesis is to expand the ECSER pipeline for the use of LLMs by adding recommendations from LLM4SE research and related fields. First, an exploratory mapping study will be done of SE studies released in 2024 that use LLM classifiers, summarising which steps are taken and which metrics are (or aren't) reported, in order to get an overview of the current state of LLM4SE research. Subsequently, we design an enhanced version of the ECSER pipeline for the use of LLMs, including recommendations for prompt engineering, evaluating the fairness and robustness of models, and more. Lastly, in order to evaluate the comprehensiveness and ease-of-use of the new pipeline, two replication studies will be conducted using the pipeline to test its ability to strengthen or contradict the findings of the original papers.

MSc. Artificial Intelligence
30EC

The Ethics and Privacy Quick Scan of the Utrecht University Research Institute of Information and Computing Sciences classified this research as low-risk with no fuller ethics review or privacy assessment required.

Contents

1 Introduction

In recent years, large language models (LLMs) have become an increasingly relevant tool in software engineering (SE), allowing researchers to accomplish complex tasks usually reserved for humans or other machine learning (ML) methods, such as code generation, summarization and classification. However, few guidelines exist for conducting and reporting on LLM research, leaving researchers on their own to decide what choices to make when designing prompts, what evaluation metrics to report in their results and the level of statistical validation needed for their comparisons. Because of these lack of guidelines, researchers using LLMs often omit important details, such as the F1 score or the significance of their results [1]. This problem is not new or unique to LLMs; many researchers have reported that there is a poor standard of empirical software engineering, which inhibits the usefulness of results and impairs reproducibility [2, 3].

To treat the observed poor standard of empirical SE, some researchers have called for a more systematic approach to conducting and evaluating experiments. The ECSER (Evaluating Classifiers in Software Engineering) pipeline, introduced by Dell'Anna et al. [4], provides a systemic pipeline for training, validating and testing ML classifiers and analysing their results. ECSER is based on recommendations from the field of machine learning for software engineering (ML4SE) and is designed to be easy to use for SE researchers, regardless of expertise with machine learning methods. The ECSER pipeline has seen some empirical validation in the form of two replication studies, in requirements engineering and software testing, where Dell'Anna et al. were able to confirm and strengthen some findings, as well as discover additional findings or findings that contradict the original ones. Figure 1 shows the steps of the ECSER pipeline.

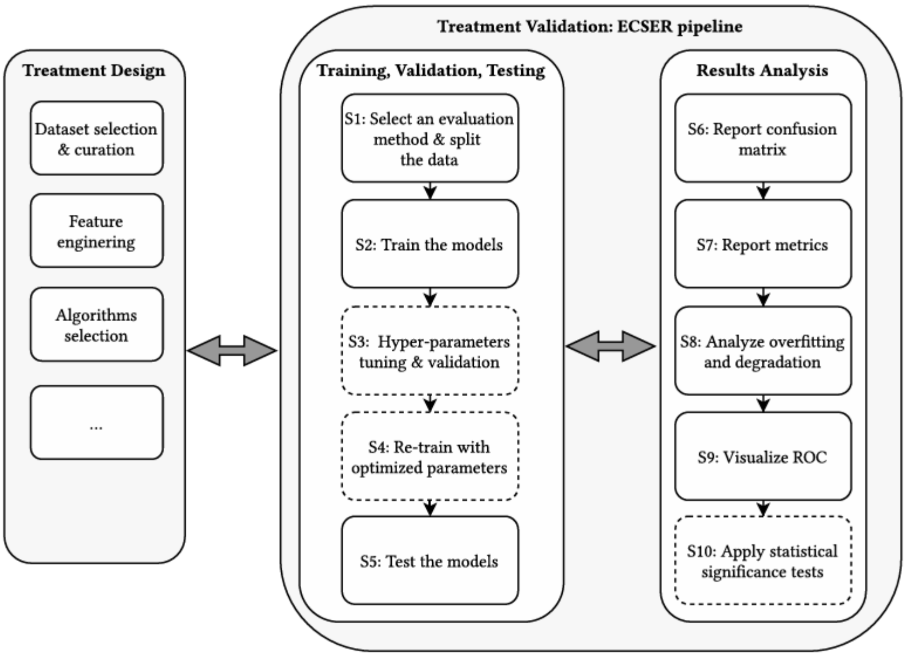

Figure 1 The ECSER pipeline. Reproduced from Dell'Anna et al. [4]. Steps with a dashed border are optional. The $\leftrightarrow$ arrows indicate feedback loops between phases

However, while ECSER is tailored to traditional ML classifiers (such as logistic regression, SVMs and decision trees), it includes no specific recommendations for LLM-based classifiers, despite LLMs showing promise for various classification tasks [5, 6]. Thus, the ECSER pipeline is not sufficient for LLM research, since using LLMs for classification comes with unique considerations that are not present for traditional classifiers. For one, it is common to use pre-trained, generalized LLMs and give task-specific instructions to get the desired classification on the input data. This means that designing these instructions, also called prompts, becomes a crucial part to conducting LLM research [7]. There are also unique considerations when it comes to evaluating the fairness and robustness of the results: does the model behave in accordance with human values and are the results resilient to adversarial prompts, such as those containing typos [8, 9]?

The goal of this thesis is to address the lack of LLM-specific guidelines in the ECSER pipeline, by expanding it to include specific recommendations for conducting LLM classifier research. In doing so, we will make the following contributions:

- We will conduct a mapping study to assess the current state of reporting in LLM4SE classification research.
- We will develop an expanded version of the ECSER pipeline that includes specific recommendations for conducting LLM classifier research, such as prompt engineering and the evaluation of model fairness and robustness.
- We will conduct two replication studies of existing LLM classifier papers using the enhanced pipeline, in order to evaluate the pipeline's usability and comprehensiveness.

The remainder of this thesis is structured as follows. In Section 2, we provide background information about LLMs. Section 3 describes the research approach and research questions. Lastly, Section 4 discusses related work.

2 Background

Language Models (LMs) are computational models designed to model the generative likelihood of word sequences [10]. In other words, given a sequence of input text, they predict the next word in the sequence based on what is deemed most likely by the model. One of the oldest examples of language models are N-grams, which model the likelihood of a word given the previous N tokens [11]. Early statistical models like N-gram models suffer from several problems, including limited context and difficulty with unseen words.

Pre-trained Language Models (PLMs) are LMs that are designed to take in vast amounts of unannotated text in order to learn knowledge about language and the world [12]. The knowledge that is contained in these models can then be used on new tasks by way of **transfer learning** [13]. PLMs require much more information to be contained within than statistical language models such as N-grams could provide, which has led researchers to design more sophisticated models that rely on neural networks to model language, starting with recurrent neural networks (RNNs) and later Long Short-Term Memory (LSTM) models [14, 15]. The biggest recent breakthrough came with the invention of **transformers**, which performed better than previous models, were easier to train and allowed for the parallel processing of inputs [16]. Transformers typically consist of an **encoder**, which embeds the input sequence into a hidden (latent) space and a **decoder**, which translates the abstract representation of the hidden space to the target text. Both the encoder and decoder use a self-attention mechanism to weigh the contribution of each token to the output [16, 17].

Large Language Models (LLMs) were introduced as a way to refer to PLMs with massive parameter sizes, since it was found that larger parameter sizes lead to better performance and emergent abilities [18, 19]. Almost all LLMs use transformer models with the self-attention mechanism, but not all of them use both the encoder and decoder component [20]:

- **Encoder-decoder LLMs**, such as BART [21] and T5X [22], are good at both understanding and generating language [20]. As such they are useful for tasks like summarization and translation [23, 24].
- **Encoder-only LLMs**, such as BERT and its variants [25–28], are tailored towards language understanding [20]. As such they are good at tasks like classification and inference [29].
- **Decoder-only LLMs**, such as the GPT-series [30–32], are tailored towards language generation [20]. As such they are good at all generation tasks, including code generation [33] and creating conversational agents like ChatGPT [34] and LLama 2 [35].

Prompt Engineering is the process of systematically designing natural language instructions (or **prompts**) to guide LLMs towards a desired output [36]. Prompts condition the outputs of the LLM on the given instructions, so that the model generates the next tokens with both the prompt and previously generated tokens in mind [12]. The most common approaches to prompt engineering are:

- **Zero-shot Prompting**. The model is given no examples for a task, forcing it to rely entirely on the prompt [37].
- **Few-shot Prompting**. A limited number of examples for a task are given, which the model can learn from [38].
- **Chain-of-Thought Prompting**. The prompt instructs the model to follow coherent reasoning steps [39].

Another less common approach is automatic prompt engineering, where instructions are automatically generated and selected [40].

3 Research Approach

The usage of LLMs in SE research has increased massively over the past few years, but the research of how to best conduct studies and report results in this field remains very limited. In order to approach the problem of LLM4SE reporting standards and design meaningful guidelines, we follow Wieringa's design cycle of *problem investigation*, *treatment design* and *treatment validation* [41]. This leads us to the following research questions:

RQ1: What is the current state of reporting in LLM4SE classification research?
The purpose of RQ1 is both descriptive and evaluative: what do researchers report in their studies and does this level of reporting contribute or detract from the accuracy and reproducibility of the results. To answer the question, we conducted a mapping study of SE papers published in 2024 that use LLM classifiers using the guidelines by Kitchenham et al. [42, 43] and Petersen et al. [44], consisting of five steps:

1. Define the research question.
2. Conduct a search for primary studies.
3. Screen papers based on inclusion/exclusion criteria.
4. Classify the papers.
5. Extract data.

What follows is a detailed summary of the search strategy (steps 2 and 3) and the information that was obtained from the list of relevant papers (steps 4 and 5).

Search Strategy

To get a complete image of the state of LLM4SE classifier evaluation, we collected all 528 papers published in 2024 in three of the top SE conferences and journals (Table 1). These venues were narrowed down from the list of venues included in [20] based on their impact and relevance. Using the full list of 528 papers, an automatic keyword search was conducted on the title and abstract of each paper to identify the papers that were potentially relevant to LLM4SE.

The list of keywords was based on previous research ([20, 45]) and designed with the intent of high recall, since it is easier to remove false positives than to retrieve false negatives. Keyword matching was done on the lowercased versions of the title and abstract and allowed for partial matches (e.g., ”llm” matches ”LLM”, ”LLMs”, ”LLM4SE”, etc.). The complete list of keywords is as follows:

- *llm, large language model, language model, code generation model, plm, pre-trained, pretrained, pre-training, pretraining, chatgpt, gpt, bert, t5, llama, bart.*

The keyword search resulted in 129 potentially relevant papers. After manually removing false positives, mostly caused by the authors comparing their non-LLM model to previous LLM models, the total number of identified LLM4SE papers was 116, representing 21.97% of total SE papers.

To narrow down the list from all LLM-related papers to those pertaining to SE classification research, the title and abstract of the remaining 116 papers were manually screened for the inclusion and exclusion criteria (Table 2). These criteria were designed to exclude any paper not relevant for answering the research question, such as meta studies and those not related to classification. During the manual search, 18 papers were identified that use LLMs for SE classification research and satisfy all other criteria.

Table 1 Publication venues included in the mapping study.

Acronym	**Venue**
ICSE	International Conference on Software Engineering
FSE	International Conference on the Foundations of Software Engineering
TSE	Transactions on Software Engineering

Table 2 Inclusion/Exclusion Criteria

Inclusion Criteria
The paper uses an LLM. The paper focuses on a downstream application of the LLM. The downstream application is a SE classification task. The full text of the paper is accessible. The paper was published in 2024.
Exclusion Criteria
The paper focuses on LLM architecture with no downstream application. The paper or task is not relevant to SE. The full text is inaccessible. The paper is a survey or other meta-analysis. The paper focuses on research methodology. The paper is a reprint or different version of another paper.

While only classification papers were included in the final analysis, all the papers that applied LLMs to SE tasks (a total of 83 papers) were additionally classified based on the type of LLM application (classification, generation or recommendation) to get a view of the relative academic interest in each type (Figure 2). As can be seen, LLMs are most often used for generation, with 73.5% of papers using them in this way, including some that use generation alongside classification. However, classification represents the second largest application of LLMs, with 22.9% in total. Appendix A contains references for Figure 2 and other pie charts contained in this thesis.

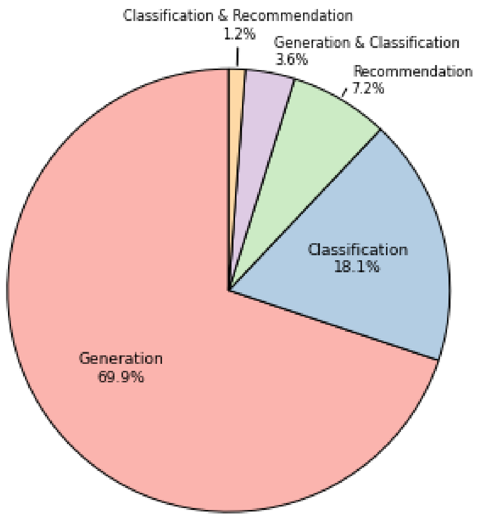

Figure 2 Distribution of LLM application types in collected studies.

Information Extraction

After completing the manual search and obtaining 18 relevant LLM4SE classification papers, the evaluation and reporting practices of each paper were analysed and recorded. In particular, we recorded whether or not the following evaluation metrics were reported: *precision, recall, accuracy, F-score, Area under curve (AUC), Matthews correlation coefficient (MCC), Confusion matrix, plot, Calibration, Fairness* and *Robustness*. Further, we looked at several factors related to the reproducibility and generalization of results, particularly whether justification was given for why the chosen evaluation metrics were used, whether results were obtained for multiple datasets, whether the results were compared to external baselines and whether the statistical significance was analysed and the choice of significance test justified.

Lastly, for the papers that used a prompt-based approach for classification, we analysed whether the prompt approach and prompt pattern were reported and justified, whether multiple prompts were compared and whether the temperatures of the models were reported.

The information that was extracted from the papers was determined based on what is recommended to be included by the ECSER pipeline [4], metrics that are recommended by other authors (such as the Matthews correlation coefficient [46–48]) and other information relevant to LLMs, like prompting technique and measures of fairness & robustness (see section 4.4 and 4.5).

RQ2: What resource can be developed to assist researchers in conducting and reporting on LLM4SE classification research?

RQ2 takes the role of treatment design; how can we solve potential problems that are identified by RQ1?

The purpose of RQ2 is to design an artifact that is easy to use and comprehensive. To answer the question, we will enhance the existing ECSER pipeline [4] to include the use of LLMs, taking into consideration recommendations from the fields of LLM4SE and ML4SE, as well as relevant statistical or methodological research in other fields (see section 4). Table 3 includes a preliminary summary of the additions that will be made to the ECSER pipeline, with the primary supporting references for the additions in brackets. The list of additions is subject to change based on the findings from RQ1 and further literature review. It is also possible that the best course of action will be to add more steps to the ECSER pipeline, as opposed to expanding existing ones, particularly for reporting the calibration, fairness and robustness or for prompt engineering.

Table 3 Preliminary summary of additions to the ECSER pipeline

ECSER Step	Additions
S1. Select an evaluation method and split the data	Specify desired level of model alignment and safety [1]
S2. Train the model	Define the goal of the prompt [7, 49] Select prompt engineering approach (zero-shot, few-shot, etc.) [20, 49] Select prompting pattern [49, 50]
S3. Hyper-parameters tuning and validation	Tune LLM model temperature [51] Compare prompting patterns [36]
S7. Report Metrics	Report Matthews correlation coefficient (MCC) [46–48] Report calibration (e.g., expected calibration error [52]) Report fairness (e.g., equalized odds difference [8]) Report robustness (e.g., performance drop rate [9])
S8. Analyse overfitting and degradation	Conduct ablation studies (optional) [53]

RQ3: How useful is the proposed pipeline in assisting LLM4SE classification research?
The purpose of RQ3 is treatment validation: does the designed pipeline in RQ2 achieve its goals of ease of use and comprehensiveness. To answer this question we will conduct two replication studies in different subfields of SE that employ LLM classification using the guidelines by Carver [54]. The original studies will be chosen based on the following criteria:

1. The authors provide a ready-to-use replication package.
2. The study is recent (past five years) and represents current practices in LLM4SE well.
3. The study is peer-reviewed and published in a reputable journal.
4. The study does not already follow all the proposed guidelines, so that the added benefit of the pipeline can be evaluated.

By conducting these replication studies, we can evaluate the comprehensiveness of the pipeline and identify potential shortcomings. In order to test the ease of use for the intended users, we aim to ask an SE researcher to use the pipeline or give their expert opinion on it if possible. After the first replication study, the findings from RQ3 will be taken into account to make changes to the guidelines from RQ2 where needed, creating a feedback-loop between RQ2 and RQ3. The second replication will be conducted using the updated pipeline and might inform further changes.

4 Related Work

The main focus of this thesis is to study the usage and evaluation of LLM Classifiers in Software Engineering, but in doing so we touch on several related research fields, including Empirical Software Engineering, Machine Learning for Software Engineering (ML4SE), Large Language Models for Software Engineering (LLM4SE) and Prompt Engineering. Additionally, each of the three research questions has related work that is relevant to their methodologies. Hence, we have split the related work into several sections based on different areas of study to aid readability.

First, we discuss challenges in empirical software engineering which motivated the creation of the ECSER pipeline and further motivate our mapping study of LLM4SE. Second, we discuss the topic of replication in SE, which is relevant to our replication studies and the field of empirical SE. Third, we discuss how Large Language Models have been used in SE, with a focus on classification tasks. Fourth, we discuss related work in Prompt Engineering, as it has become a vital part to using LLMs and will inform our guidelines for prompt engineering in the enhanced ECSER pipeline. Finally, we discuss related work on the evaluation of LLMs and classifiers which will inform our guidelines for the evaluation of LLM Classifiers.

4.1 Challenges in Empirical Software Engineering

Over the past couple decades, the use of machine learning methods has become ubiquitous in every field of science, among them software engineering. Zhang and Tsai [55] summarized seven main activities that use machine learning in SE: prediction, property/model discovery, transformation, generation, library construction and maintenance, acquisition of specifications and development knowledge management. In addition, they list specific tasks belonging to each activity, such as *predicting* software quality and *generating* test cases. The field of classification research falls almost exclusively in the prediction category.

With the increased popularity of machine learning methods, many software packages have been developed to allow anyone to use these methods without having to write any code or even understand how these methods work. This leads to many researchers using machine learning without fully understanding the underlying assumptions of the methods they are using and assuming the results are correct without proper (statistical) analysis. This problem is not new: Kitchenham et al. [2] reported at the start of this century that there was a poor standard of empirical software engineering, mostly caused by a lack of statistical expertise from both the authors and reviewers. They go on to provide extensive guidelines for designing and conducting experiments as well as analysing, presenting and interpreting results. In another paper, Kitchenham et al. [56] discuss the benefits of evidence-based software engineering approach by drawing an analogy to evidence-based medicine, but note that the infrastructure needed for widespread adoption of evidence-based SE isn't there yet, which means that SE experiments are vulnerable to subject and experimenter bias.

Methodological problems in SE research can result in problems with the reproducibility of results, which severely reduces the real-world usefulness of said results. Menzies and Shepperd [3] argue that the main goal of science is *conclusion stability*, which means that when you discover an effect, it holds true irrespective of the situation. However, they note that in the software engineering literature there is often as much evidence *for* a given effect, as *against*. This lack of conclusion stability is caused by sources of bias and variance that are introduced at various steps: sampling, pre-processing, training, algorithm selection and the analysis of results. If these sources of bias and variance are accurately analysed and reported, that would make it easier for the authors and other researchers to interpret the conclusions.

Despite being a known issue, the problems of improper analysis and reporting have persisted over the years. In a mapping study of the proceedings of the International Conference of Software Engineering (ICSE) from 2019 to 2021, Dell'Anna et al. [4] analysed 60 SE papers related to classification and found that most papers

(36) reported on the precision and recall of their experiment, fewer reported the F-score (27) and accuracy (24), but only 14 papers explicitly justified why they chose the reported metrics. The full confusion matrix was only included in six papers. Other metrics like the receiver operator characteristic (ROC) curve and the area under the curve (AUC) were mostly absent with 3 and 9 respective mentions. Of course, as we know from statistics, it is not enough to report a metric and claim it's better than that of other models, since it could be coincidental, so statistical significance testing is required to make such claims. However, only six papers analysed the statistical significance of their results. Evidently, the situation is dire, which is why Dell'Anna et al. developed the ECSER pipeline to give SE researchers a systematic approach to conducting classification experiments and reporting and analysing the results.

4.2 Replications in SE

Replication studies are a vital part of empirical software engineering, allowing researchers to investigate the effects of alternative values for important attributes, vary the strategy with which hypotheses are investigated and make up for certain threats to validity [57]. The importance of replication is not unique to software engineering, after all the so-called "replication crisis" was a massive topic of debate in psychology research in recent years, since it was found that many previous results in the field could not be replicated, even if the original findings were statistically significant [58–60]. Cruz et al. [61] conducted a systematic literature review of replication studies in SE between 2013 and 2018 and found that the number of published replication studies has been steadily increasing, though there are research gaps in certain fields, suggesting that the SE community is taking an increased interest in replicating previous studies.

Shull et al. [62] identify two types of replication studies: *exact replications*, in which the procedures of the original study are strictly followed; and *conceptual replications*, in which the research problem of the original study is evaluated using a different procedure. Additionally, exact replications are subdivided into *dependent* and *independent* replications. Dependent replications keep most of the conditions of the experiment (close to) the same, which allows one to test the effect of specific variables on the results; whereas independent replications vary the original conditions of the experiment significantly, which allows one to test the robustness of the original results under different conditions. Juristo and Vegas [63] underline the value of non-exact replications, which they define similarly to how Shull et al. define independent exact replications, by conducting a multiple-case replication study using a newly proposed four-phase process, to show that non-identical replications can be used to learn new information about the original results.

While there is a consensus in SE that replication is important, Shepperd et al. [64] show that replication studies often suffer from the same problems as original studies, such as incomplete reporting, low power or potential researcher bias. Notably, many replication studies in their survey did not report any details on the dispersion (e.g., variance) of the response variables. In another paper, Shepperd [65] shows that because of wide prediction intervals, almost all replications are confirmatory, which means the added knowledge is negligible. Because of these problems, they suggest that researchers should focus more on broader meta-analyses instead of replication studies. Other common problems in conducting replication research include difficulty in getting them published, lack of guidelines and the unavailability of replication packages [61, 66].

Communication can also play a factor in successful replication: Vegas et al. [67] investigated the role of communication in successful replication of previous experiments and found that there should be some level of communication, orally or in writing, between the previous authors and authors of the replication study, to avoid unnecessary changes to the experiments.

Given these challenges, guidelines for conducting replications are clearly needed. Despite this, the number of existing guidelines remains low. Guidelines were proposed by Carver [54], which are still commonly used in replications, but these seem to be the only replication guidelines for software engineering and were intended as a starting point for future guidelines. Other artifacts do exist to aid replications: Gómez et al. [68] proposed a classification of replications in SE, differentiating between literal, operational and conceptual replications, that can be used to identify which changes can be made in each type of replication and understand the level or verification needed for that type.

Abualhaija et al. [69] proposed an artifact, referred to as the ID-Card, for summarizing papers in RE research that use NLP techniques. The intention behind the ID-Card is that all the relevant information needed for replication is presented in an easy to read format and can be used for both replication and educational purposes. Lastly, Brandt et al. [70] developed the "Replication Recipe" for psychology research, which lists 36 questions that should be addressed when conducting a replication, most of which are also directly applicable to empirical SE.

4.3 Large Language Models for SE

A recent development in SE research is the increased relevance and use of LLMs. Fan et al. [71] looked at all preprints on arXiv categorised under computer science whose title included "LLM", "Large Language Model" or "GPT" and found the number of SE papers that mentioned LLMs grew exponentially from 0 in 2019, 5 in 2020, to 181 in 2023. Naturally, this is only an approximation of the total number of preprints on arXiv that use LLMs, but it undoubtedly signifies a substantial increase. Fan et al. also conducted an extensive survey of how LLMs have been applied to various software development activities (such as code generation and testing) and research domains (such as human-computer interaction and education), but the survey is almost exclusively focused on the generative use of LLMs and does not go into depth on the use of LLMs in classification research. Many other good surveys and analyses of the use of LLMs in Software Engineering have been written, but most of them similarly do not mention any applications to classification tasks and instead exclusively focus on generation [72–74].

Surveys have also been written about specific domains of software engineering: Wang et al. [75] did a comprehensive review of 102 studies that use LLMs for software testing and found that LLMs have commonly been applied to various tasks, including test case generation, test input generation, debugging, program repair and more. They also found that while most papers used LLMs to address the entire task, many others combined LLMs with additional techniques to optimize the outputs of the LLM, including mutation testing, differential testing, syntactic checking, program analysis, statistical analysis and other techniques. With these extra techniques, researchers were able to generate more diverse and complex code and overcome some of the limitations of LLMs.

While LLMs are most frequently used to generate text (and code), they have seen considerable success in classification tasks. Guo et al. [5] have shown that LLMs can outperform other methods like SVMs for health-related text classification. Fields et al. [6] present an in-depth survey of text classification using transformers across domains and found that LLMs can perform remarkably well on many (but not all) classification tasks. However, Chen et al. [77] compared the performance of various LLMs to traditional ML methods in clinical prediction and found that LLMs could *not* beat traditional methods in this case. Vajjala and Shimangaud [78] also show that there are large performance disparities between languages in classification tasks. Thus, LLMs might not always be a better choice than traditional ML methods, but they certainly show potential.

Software engineers have also started using LLMs for classification. Hou et al. [20] conducted a systematic literature review of 395 software engineering studies that use LLMs and found that around 21.61% involve classification tasks. The most common classification tasks where LLMs have been used include vulnerability detection, requirements classification, bug prediction and review/commit/code classification, with many other classification tasks having been attempted using LLMs. Zhang et al. [45] conducted a systematic survey of 947 SE studies and summarized 62 unique LLMs of code, including six LLMs specifically fine-tuned for the tasks of code classification, clone detection and defect detection.

Recently, Baltes et al. [76] presented a list of community-driven guidelines for using and reporting on LLMs, providing eight detailed guidelines. These guidelines include: (1) declaring LLM usage and role; (2) reporting model versions, configurations and fine-tuning; (3) documenting tool architectures; (4) disclosing prompts and interaction logs; (5) using human validation; (6) employing an open LLM as baseline; (7) using suitable baselines, benchmarks and metrics; and (8) openly articulating limitations and mitigations. These guidelines were designed to increase the transparency and reproducibility of LLM research and are actively maintained on `https://llm-guidelines.org/`. The guidelines in this thesis serve a similar purpose of helping researchers conduct LLM4SE research; however, their guidelines have a broader focus, since they include all LLM applications, whereas our guidelines are specifically tailored to classification research. Furthermore, unlike their paper, this thesis also includes a mapping study to assess the state of LLM4SE classification reporting and provides a replication study to evaluate the guidelines. Despite the difference in scope, many of their recommendations directly apply to our research and were taken into account for the enhanced ECSER pipeline where relevant.

4.4 Prompt Engineering

Designing a good prompt for a particular task is vital to ensuring the adequate performance of LLMs, which is why much research has gone into the systematic designing of prompts, or prompt engineering. Marvin et al. [7] provide an overview of prompt engineering principles and techniques and outline the main steps involved in the process of prompt engineering: 1) define the goal of the prompt; 2) understand the model's capabilities; 3) choose the right prompting format; 4) provide context to the LLM and 5) test and refine the prompt based on the goal. Several resources exist to make it easier to choose the right prompting approach: White et al. [50] introduce prompt patterns, which offer reusable solutions to common prompting challenges, and provide a framework for designing and documenting these patterns in terms of the intent, motivation, structure and consequences of the patterns. For example, the persona pattern can be used to make the LLM take the point of view of an expert in the field, which is especially useful if the user itself is not an expert. The paper provides a catalogue of this pattern and other successfully applied patterns with example implementations. Further, Ekin [79] provides an accessible (AI-generated) guide to prompt engineering.

Prompt engineering is an important consideration when using LLMs: Sclar et al. [80] show that choices in prompt design, even if minor, can have a very large impact on the performance of several popular LLMs, and this sensitivity is present regardless of the size of the model or the number of examples provided. Because of this, they argue that researchers should forego using only a single prompt and switch to reporting the performance of their models using multiple plausible prompt patterns. They present a novel algorithm, FormatSpread, to evaluate multiple prompts at the same time and measure the sensitivity of the LLM. Mizrahi et al. [81] have found a similar sensitivity of LLMs to prompt design and discuss metrics to evaluate multiple prompts. Lastly, Maia Polo et al. [82] propose PromptEval, an efficient method for evaluating a large number of prompts, and show it can accurately estimate the performance distribution of the prompts.

Prompt engineering research has also been conducted specifically for software engineering: Hou et al. [20] looked at what prompt engineering techniques are commonly applied in SE tasks and found the most common techniques involve few-shot prompting and zero-shot prompting, but the third largest group of studies had no explicit mention of prompting techniques or proposed their own strategies. Other techniques included chain-of-thought, automatic prompt engineering, chain-of-code, automatic chain-of-thought, modular-of-thought and structured chain-of-thought.

Arvidsson and Axell [49] collected prompt engineering recommendations for requirements engineering (RE) in a systematic literature review and classified the guidelines into various themes, after which they interviewed three RE experts to evaluate the guidelines. Ronanki et al. [36] evaluated the effectiveness of 5 prompting patterns (from the catalogue by White et al. [50]) on two RE tasks and proposed a framework for evaluating the effectiveness of prompting patterns for any RE task, providing five steps for conducting a comparison of patterns.

A recent development in prompt engineering is automatic prompt engineering (APE), which has shown good performance compared to baseline models, while avoiding the need for manually designing prompts [40, 83]. Zadenoori et al. [84] investigated the use of automatic prompt engineering in RE and found that, on average, APE outperforms traditional prompt engineering techniques. However, research is still limited.

4.5 Evaluating LLM Classifiers

Few papers have been written specifically about the evaluation of LLM-based classifiers, but we can look at the evaluation of classifiers and LLMs separately to get a picture. The ECSER pipeline is the most relevant for evaluating classifiers, since it gives in-depth recommendations about the conducting and reporting of classifier research [4]. Specific recommendations include the reporting of the full confusion matrix, reporting other metrics relevant to the domain such as specificity (true negative rate), analysing overfitting and degradation, visualising the ROC and applying statistical significance tests. Some researchers also recommend reporting the Matthews correlation coefficient (MCC), which combines all four metrics of the confusion matrix and can be more informative than the F1 score [46–48].

Hou et al. [20] looked at whether these metrics were reported in LLM-based classification research and found that out of 147 papers that use LLMs for classification, the most frequently reported metric is precision with 35 papers, followed by recall (34), F1-score (33) and accuracy (23), but the AUC was reported 9 times, the ROC only 4 times and the MCC only twice. Unfortunately, they did not count how many studies did significance testing. These results are very similar to Dell'Anna et al. [4]'s aforementioned mapping study of classifier research as a whole and suggest that many important metrics are under-reported in LLM-based classification research.

Using and evaluating LLMs comes with more considerations than just the accuracy of the results. Guo et al. [1] categorize the evaluation of LLMs into three types: knowledge and capability evaluation, which assesses the fundamental knowledge and reasoning capabilities of the LLM; alignment evaluation, which refers to evaluating ethical and moral considerations like bias; and safety evaluation, which focuses on the robustness of LLMs and risk evaluation. Liu et al. [85] have created guidelines specifically for evaluating LLM alignment in terms of reliability, safety, fairness, resistance to misuse, explainability and reasoning, adherence to social norms and robustness.

Another consideration is how well the LLM outputs adhere to the format specified in the prompt. For example, it is common to ask an LLM to format its answer as a valid JSON object, allowing for easier processing of the output. Limiting responses to a specified format can be beneficial: Tam et al. [86] compared the performance of several LLMs between free-form response and formatted responses and found that, while reasoning ability was weakened for formatted responses, classification accuracy was increased. However, LLMs do not always follow the format accurately. Long et al. [87] define several measures to analyse the level of adherence to the output format: SysE, which measures the performance of the LLM for answers that *meet the constraint*; TrueE, which measures the performance of all answers *regardless of whether the format is satisfied*; and BiasF, which measures the (mean squared) difference between SysE and TrueE. Li et al. [88] introduce the JScore measure, which measures the similarity between JSON objects and can be used to compare LLM generated JSON objects with the target JSON objects. Lastly, Xia et al. [89] introduce the FoFo benchmark for evaluating LLMs based on their ability to follow complex, domain-specific formats.

Chang et al. [90] conducted another survey of the evaluation of LLMs which focuses on downstream applications, resulting in a classification of four aspects of LLM evaluation: the accuracy, containing measures of correctness such as the F1 score; the calibration, which contains measures for the degree of alignment between the predicted probabilities and actual probabilities, such as the expected calibration error (ECE); the fairness, with measures pertaining to the equal treatment of different groups, such as the equalized odds difference (EOD); and robustness, with measures pertaining to the performance of a model in the face of challenging inputs and noise, such as performance drop rate (PDR). Further, they mention several ways in which human evaluation can be used to evaluate LLMs, such as the degree to which the model outputs align with human values.

Lastly, the model temperature, which regulates the randomness of the model, can also affect its performance. Peeperkorn et al. [51] looked at the effect of model temperature on the level of creativity of the outputs and found that LLMs generate slightly more novel outputs with higher temperature, but the temperature was also correlated with incoherence. No relationship was found between temperature and cohesion or typicality. Model temperature might also affect a model's susceptibility to security attacks: Yu et al. [91] found that some LLMs became more susceptible to jailbreaking attacks as temperature increased, whereas others had decreased susceptibility to jailbreaking with increased temperature, possibly due to the fact that the former models had a lower susceptibility at 0 temperature and the latter models had a higher susceptibility at 0 temperature.

5 RQ1: What is the current state of reporting in LLM4SE classification research?

In this section, we first discuss the results obtained from the mapping study of LLM4SE classification papers released in 2024 and subsequently provide an answer to RQ1.

Results
During the search process described in section 3, 18 papers were identified as relevant for answering the research question. Firstly, we will look at the distribution of the included papers. All three venues included in the search strategy are represented in the final selection, with eight ICSE papers, eight TSE papers and two FSE papers (Figure 3).

Some of the data extraction was dependent on whether or not the papers used a prompt-based approach to LLM classification or another approach such as fine-tuning. Figure 4 shows that 44.4% or 8 papers used a prompt-based approach, while 55.6% or 10 papers used a different approach such as fine-tuning. Note that additional references for the figures and the subsequent tables are included in Appendix A.

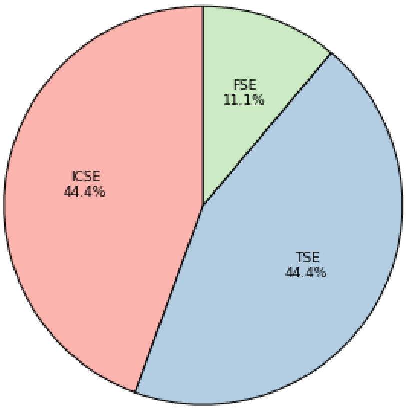

Figure 3 Distribution of venues in mapping study.

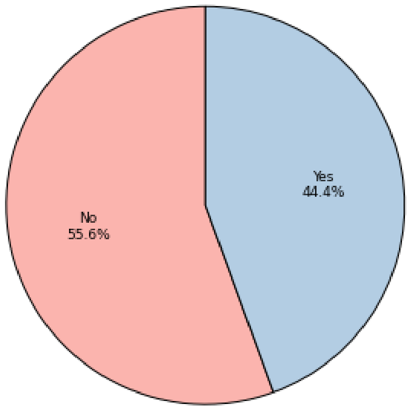

Figure 4 Distribution of prompt usage in mapping study.

Table 4 Summary of mapping study results.

Category	**Total**
Total mapping study papers	18
Evaluation metrics	
Precision	17 (94.4%)
Recall	17 (94.4%)
Accuracy	11 (61.1%)
F-Score	17 (94.4%)
AUC	3 (16.7%)
ROC plots	0 (0.0%)
MCC	0 (0.0%)
Calibration	0 (0.0%)
Fairness	0 (0.0%)
Robustness	
Data distribution & Adversarial Attacks	1 (5.5%)
Noisy data	1 (5.5%)
Metrics justification	
No	9 (50.0%)
Implicit	0 (0.0%)
Previous work	5 (27.8%)
Yes	4 (22.2%)
Confusion matrix	4 (22.2%)
Evaluation over multiple data sets	5 (27.8%)
Type of baseline	
None	2 (11.1%)
Own	3 (16.7%)
External	4 (22.2%)
External and Own	9 (50.0%)
Analysis of statistical significance	5 (27.8%)
Statistical significance justification	
No	4 (80.0%)
Implicit	0 (0.0%)
Previous work	1 (20.0%)
Yes	0 (0.0%)

Table 4 shows a summary of the results of the mapping study. Starting with the evaluation metrics that are commonly recommended in the field, of the 18 included papers, 17 reported the precision, recall and F-score of the results, while only 11 papers reported the accuracy and only four papers reported the full confusion matrix. While the precision, recall and F-score are often more informative for the usefulness of a model compared to the accuracy, by not providing the accuracy or the full confusion matrix, relevant information is lost that cannot be obtained from just three metrics, which prevents further comparisons between models that do provide the full matrix. Further, the Matthews Correlation Coefficient (MCC), which some consider more balanced than the F-score, since it takes the full confusion matrix into account [46], was not reported in any of the papers.

Unfortunately. only three papers reported the area under the curve (AUC) of the receiver operating characteristic (ROC) curve, while none provided a plot of the full ROC curve itself. This is despite the fact that AUC has been theoretically and empirically shown to be a better metric for comparing classification models than the accuracy [92, 93]. Since the AUC is a summary of the ROC curve, the full curve should also be provided for a more detailed view of the discriminatory performance of the model [94].

Further, it is recommended to measure the calibration, fairness and robustness when using LLMs. However, none of the papers reported the calibration and fairness, whereas only two papers include an analysis of the robustness. Since none of the included papers involve human participants or sensitive data, the omission of evaluating fairness can be justifiable, although it can still provide insight into the relative performance of a model on different data classes. However, the calibration should generally be analysed to evaluate the reliability of a model, since the calibration evaluates how well the predicted probabilities align with actual probabilities [95, 96].

The robustness, which evaluates the performance of a model in the face on challenging inputs, can be observed in several ways, including by making changes in the data distribution, the addition of noise to the data, and adversarial attacks [90]. Only two papers did testing of the robustness of their model in these ways, one of which looked at both altering the data distribution as well as adversarial attacks and the other looked at the effect of noisy data. However, some papers tested something similar to robustness by performing ablation studies, which work by removing parts of the model to see the impact on the performance, which allows one to assess whether the model exhibits graceful degradation [53]. Ablation studies were not initially included in the mapping study but after seeing their successful implementation were added to the enhanced ECSER pipeline as an optional step.

The presence or omission of evaluation metrics should be justified in some way, as well as the relative importance for answering the research question. For example, including or omitting the accuracy could be motivated by referring to previous work or an explanation as to why it is or isn't important for the given task. Additionally, the precision or recall can be argued to be more important for the given task than the other. However, only nine papers included any justification for their choice of metrics and only four of those provided an explanation for their choice instead of referring to previous research. Referring to previous research is better than no explanation but it runs the risk of bad habits propagating through the field.

Lastly, we looked at other information about the evaluation and comparison of models. Only five papers evaluated their models over multiple data sets, which is crucial for avoiding overfitting. Further, 13 papers compared their results to external baselines (9 of which compared to both external and their own models), though three papers only compared the results to their own baselines and two papers made no comparisons. However, despite almost all papers comparing their results to their own or other research and usually making explicit claims about the superior performance of their models, only five papers did any statistical significance testing on their results, without which statistical claims about relative performance are completely unsubstantiated. Some papers even claimed their results show a *significant* improvement despite not doing or not reporting on the significance testing. It should also be noted that of the five papers that did a statistical analysis of the results, only one justified the statistical test that was chosen by referring to previous research. However, since there are often multiple statistical tests that can be applied with their own positives and negatives, it is better to explicitly justify the choice. It is further important to justify the significance threshold instead of relying on commonly used values, since the consequences of errors depend on the application. For example, the consequences of false positives might be worse in a medical context compared to a software engineering context.

Table 5 Summary of prompt reporting results in the mapping study.

Category	Total
Total prompt-based classification papers	8
Prompt approach	
Chain-of-verification	1 (12.5%)
Few-shot	2 (25.0%)
Mimic-in-the-background	1 (12.5%)
Not reported	1 (12.5%)
Zero-shot	3 (37.5%)
Prompt pattern	
Not reported	2 (25.0%)
Own	6 (75.0%)
Prompt approach justification	
No	5 (62.5%)
Yes	3 (37.5%)
Prompt design justification	
No	4 (50.0%)
Previous Work	1 (12.5%)
Yes	3 (37.5%)
Evaluation over multiple prompts	
No	8 (100.0%)
Temperature	
No	7 (87.5%)
Yes	1 (12.5%)

Some of the papers, eight to be exact, used prompting to obtain results from pre-trained LLMs, which comes with additional considerations and challenges, such as what prompt approach to use (few-shot, zero-shot, etc.) and how to effectively write the text of the prompt. Table 5 shows a summary of the reported prompting information of the papers that use prompts in the mapping study.

Firstly, papers used various prompt design approaches; one used chain-of-verification, two used few-shot, one used a novel approach called mimic-in-the-background, three used zero-shot prompting, and one paper failed to report the prompt approach used or didn't use a systematic approach. Naturally, there is no single correct prompt approach as long as the choice is motivated well, since the performance of a prompt might depend on the limitations of the LLM or the application [7]. Unfortunately, only three papers provided justification for the chosen prompt approach.

Secondly, the prompt pattern, which is the actual prompt text used with placeholder values for data insertion, was only reported in six out of eight papers. Not providing the actual prompts prevents other researchers from conducting exact replications of the research. Further, the authors of the six papers that provided the prompt pattern wrote the prompts themselves without consulting established patterns, which means there is no prior evidence of the efficacy of the prompt patterns. Additionally, only four papers provided a justification for how they designed the prompt pattern, one of which only referred to previous work.

The performance of LLMs can be heavily impacted by minor changes in the prompt design, which is why it is recommended to *always* compare multiple plausible prompts [80]. However, none of the papers evaluated the performance of their models using multiple prompts, which leaves the vulnerability of the models to prompt variability unknowable. Lastly, only one paper reported the temperature used for the LLM, despite this parameter having a potential impact on the performance of LLMs [51].

Conclusion
Unfortunately, the results confirm a pattern of incomplete reporting and unsubstantiated claims similar to that found previously in the fields of ML4SE [4] and LLM4SE [20]. Metrics such as the AUC, ROC plots, the MCC, the full confusion matrix and metrics related to the calibration, fairness and robustness of LLMs go completely or almost completely unreported. Even when metrics are reported, there is often no explanation as to the why the metrics were chosen for the specific tasks. Further, many papers only use one data set and sometimes make no comparisons to external baselines. Worst of all, many papers make unsubstantiated claims about the relative performance of their models without significant statistical backing and when significance testing is performed, little explanation is given to the choice of test and significance threshold.

For papers that use prompts for LLM classification, there is often little to no explanation for the chosen prompt approach and the design of the prompt pattern. Further, prompt patterns are generally not based on established patterns with empirical backing, but designed without a systematic approach. Lastly, none of the papers evaluated multiple prompts to avoid the potential sensitivity of LLMs to minor changes in wording, and the temperature of LLMs goes largely unreported as well.

Naturally, although multiple venues were considered and all papers from 2024 were considered from these venues, these results are still based on a limited set of papers. However, the results align with findings from a large base of previous research [1–4, 20], so it is not unexpected to find similar problems here, especially since there are still very few guidelines for conducting LLM research.

In conclusion, there is a clear need for easy to use and comprehensive guidelines to aid researchers in conducting LLM4SE classification research, especially for statistical significance testing and prompt engineering. We designed the enhanced ECSER pipeline (section 6) to take into account the mentioned common shortcomings of existing research and provide extra attention to these topics.

6 RQ2

References

[1] Z. Guo et al., *Evaluating large language models: A comprehensive survey*, 2023. arXiv: `2310.19736 [cs.CL]`. [Online]. Available: `https://arxiv.org/abs/2310.19736`

[2] B. Kitchenham et al., "Preliminary guidelines for empirical research in software engineering," *IEEE Transactions on Software Engineering*, vol. 28, no. 8, pp. 721–734, 2002. DOI: `10.1109/TSE.2002.1027796`

[3] T. Menzies and M. Shepperd, "Special issue on repeatable results in software engineering prediction," *Empirical Software Engineering*, vol. 17, no. 1–2, pp. 1–17, Jan. 2012, ISSN: 1573-7616. DOI: `10.1007/s10664-011-9193-5` [Online]. Available: `http://dx.doi.org/10.1007/s10664-011-9193-5`

[4] D. Dell'Anna, F. B. Aydemir, and F. Dalpiaz, "Evaluating classifiers in se research: The ecser pipeline and two replication studies," *Empirical Softw. Engg.*, vol. 28, no. 1, Nov. 2022, ISSN: 1382-3256. DOI: `10.1007/s10664-022-10243-1` [Online]. Available: `https://doi.org/10.1007/s10664-022-10243-1`

[5] Y. Guo, A. Ovadje, M. Al-garadi, and A. Sarker, "Evaluating large language models for health-related text classification tasks with public social media data," *Journal of the American Medical Informatics Association : JAMIA*, vol. 31, pp. 2181–2189, 2024. DOI: `10.1093/jamia/ocae210`

[6] J. Fields, K. Chovanec, and P. Madiraju, "A survey of text classification with transformers: How wide? how large? how long? how accurate? how expensive? how safe?" *IEEE Access*, vol. 12, pp. 6518–6531, 2024. DOI: `10.1109/ACCESS.2024.3349952`

[7] G. Marvin, N. Hellen, D. Jjingo, and J. Nakatumba-Nabende, "Prompt engineering in large language models," in *Data Intelligence and Cognitive Informatics*, I. J. Jacob, S. Piramuthu, and P. Falkowski-Gilski, Eds., Singapore: Springer Nature Singapore, 2024, pp. 387–402, ISBN: 978-981-99-7962-2.

[8] B. Woodworth, S. Gunasekar, M. I. Ohannessian, and N. Srebro, "Learning non-discriminatory predictors," in *Proceedings of the 2017 Conference on Learning Theory*, S. Kale and O. Shamir, Eds., ser. Proceedings of Machine Learning Research, vol. 65, PMLR, Jul. 2017, pp. 1920–1953. [Online]. Available: `https://proceedings.mlr.press/v65/woodworth17a.html`

[9] K. Zhu et al., *Promptrobust: Towards evaluating the robustness of large language models on adversarial prompts*, 2024. arXiv: `2306.04528 [cs.CL]`. [Online]. Available: `https://arxiv.org/abs/2306.04528`

[10] W. X. Zhao et al., *A survey of large language models*, 2025. arXiv: `2303.18223 [cs.CL]`. [Online]. Available: `https://arxiv.org/abs/2303.18223`

[11] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948. DOI: `10.1002/j.1538-7305.1948.tb01338.x`

[12] D. Jurafsky and J. H. Martin, "Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition with language models," Online manuscript released January 12, 2025, 2025. [Online]. Available: `https://web.stanford.edu/~jurafsky/slp3/`

[13] H. Wang, J. Li, H. Wu, E. Hovy, and Y. Sun, "Pre-trained language models and their applications," *Engineering*, vol. 25, pp. 51–65, 2023, ISSN: 2095-8099. DOI: `https://doi.org/10.1016/j.eng.2022.04.024` [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S2095809922006324`

[14] T. Mikolov and G. Zweig, "Context dependent recurrent neural network language model," in *2012 IEEE Spoken Language Technology Workshop (SLT)*, IEEE, 2012, pp. 234–239.

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[16] A. Vaswani et al., *Attention is all you need*, 2017. DOI: 10.48550/ARXIV.1706.03762 [Online]. Available: https://arxiv.org/abs/1706.03762

[17] D. Bahdanau, K. Cho, and Y. Bengio, *Neural machine translation by jointly learning to align and translate*, 2016. arXiv: 1409.0473 [cs.CL]. [Online]. Available: https://arxiv.org/abs/1409.0473

[18] J. Kaplan et al., *Scaling laws for neural language models*, 2020. arXiv: 2001.08361 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2001.08361

[19] J. Wei et al., *Emergent abilities of large language models*, 2022. arXiv: 2206.07682 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2206.07682

[20] X. Hou et al., *Large language models for software engineering: A systematic literature review*, 2024. arXiv: 2308.10620 [cs.SE]. [Online]. Available: https://arxiv.org/abs/2308.10620

[21] M. Lewis et al., *Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension*, 2019. arXiv: 1910.13461 [cs.CL]. [Online]. Available: https://arxiv.org/abs/1910.13461

[22] A. Roberts et al., "Scaling up models and data with t5x and seqio," *arXiv preprint arXiv:2203.17189*, 2022. [Online]. Available: https://arxiv.org/abs/2203.17189

[23] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, *On the properties of neural machine translation: Encoder-decoder approaches*, 2014. arXiv: 1409.1259 [cs.CL]. [Online]. Available: https://arxiv.org/abs/1409.1259

[24] A. Asadi and R. Safabakhsh, "The encoder-decoder framework and its applications," in *Deep Learning: Concepts and Architectures*, W. Pedrycz and S.-M. Chen, Eds. Cham: Springer International Publishing, 2020, pp. 133–167, ISBN: 978-3-030-31756-0. DOI: 10.1007/978-3-030-31756-0_5 [Online]. Available: https://doi.org/10.1007/978-3-030-31756-0_5

[25] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, 2019. arXiv: 1810.04805 [cs.CL]. [Online]. Available: https://arxiv.org/abs/1810.04805

[26] Y. Liu et al., *Roberta: A robustly optimized bert pretraining approach*, 2019. arXiv: 1907.11692 [cs.CL]. [Online]. Available: https://arxiv.org/abs/1907.11692

[27] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: Smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.

[28] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, *Albert: A lite bert for self-supervised learning of language representations*, 2020. arXiv: 1909.11942 [cs.CL]. [Online]. Available: https://arxiv.org/abs/1909.11942

[29] M. V. Koroteev, *Bert: A review of applications in natural language processing and understanding*, 2021. arXiv: 2103.11943 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2103.11943

[30] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al., "Improving language understanding by generative pre-training," 2018.

[31] T. B. Brown et al., *Language models are few-shot learners*, 2020. arXiv: 2005.14165 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2005.14165

[32] OpenAI et al., *Gpt-4 technical report*, 2024. arXiv: 2303.08774 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2303.08774

[33] R. A. Poldrack, T. Lu, and G. Beguš, *Ai-assisted coding: Experiments with gpt-4*, 2023. arXiv: 2304.13187 [cs.AI]. [Online]. Available: https://arxiv.org/abs/2304.13187

[34] OpenAI, *Chatgpt*, https://chat.openai.com/, Available at https://chat.openai.com/, 2023.

[35] H. Touvron et al., *Llama 2: Open foundation and fine-tuned chat models*, 2023. arXiv: 2307.09288 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2307.09288

[36] K. Ronanki, B. Cabrero-Daniel, J. Horkoff, and C. Berger, *Requirements engineering using generative ai: Prompts and prompting patterns*, 2023. arXiv: 2311.03832 [cs.SE]. [Online]. Available: https://arxiv.org/abs/2311.03832

[37] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al., "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[38] T. Brown et al., "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[39] J. Wei et al., "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems*, vol. 35, pp. 24 824–24 837, 2022.

[40] Y. Zhou et al., *Large language models are human-level prompt engineers*, 2023. arXiv: 2211.01910 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2211.01910

[41] R. J. Wieringa, *Design Science Methodology for Information Systems and Software Engineering*. Springer Berlin Heidelberg, 2014, ISBN: 9783662438398. DOI: 10.1007/978-3-662-43839-8 [Online]. Available: http://dx.doi.org/10.1007/978-3-662-43839-8

[42] B. A. Kitchenham, D. Budgen, and O. Pearl Brereton, "Using mapping studies as the basis for further research – a participant-observer case study," *Information and Software Technology*, vol. 53, no. 6, pp. 638–651, 2011, Special Section: Best papers from the APSEC, ISSN: 0950-5849. DOI: https://doi.org/10.1016/j.infsof.2010.12.011 [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584910002272

[43] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," vol. 2, Jan. 2007.

[44] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*, vol. 17, Jun. 2008.

[45] Q. Zhang et al., *A survey on large language models for software engineering*, 2024. arXiv: 2312.15223 [cs.SE]. [Online]. Available: https://arxiv.org/abs/2312.15223

[46] D. Chicco and G. Jurman, "The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation," *BMC Genomics*, vol. 21, no. 1, Jan. 2020, ISSN: 1471-2164. DOI: 10.1186/s12864-019-6413-7 [Online]. Available: http://dx.doi.org/10.1186/s12864-019-6413-7

[47] G. M. Foody, "Challenges in the real world use of classification accuracy metrics: From recall and precision to the matthews correlation coefficient," *PLOS ONE*, vol. 18, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:263668955

[48] J. Yao and M. J. Shepperd, "Assessing software defection prediction performance: Why using the matthews correlation coefficient matters," *Proceedings of the 24th International Conference on Evaluation and Assessment in Software Engineering*, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:211818318

[49] S. Arvidsson and J. Axell, "Prompt engineering guidelines for llms in requirements engineering," 2023.

[50] J. White et al., *A prompt pattern catalog to enhance prompt engineering with chatgpt*, 2023. arXiv: 2302.11382 [cs.SE]. [Online]. Available: https://arxiv.org/abs/2302.11382

[51] M. Peeperkorn, T. Kouwenhoven, D. Brown, and A. Jordanous, *Is temperature the creativity parameter of large language models?* 2024. arXiv: 2405.00492 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2405.00492

[52] M. Pakdaman Naeini, G. Cooper, and M. Hauskrecht, "Obtaining well calibrated probabilities using bayesian binning," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, no. 1, Feb. 2015. DOI: 10.1609/aaai.v29i1.9602 [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/9602

[53] R. Meyes, M. Lu, C. W. De Puiseau, and T. Meisen, "Ablation studies in artificial neural networks," *arXiv preprint arXiv:1901.08644*, 2019.

[54] J. C. Carver, "Towards reporting guidelines for experimental replications: A proposal," in *Proceedings of the 1st International Workshop on Replication in Empirical Software Engineering Research (RESER)*, Held during ICSE 2010, Cape Town, South Africa, May 2010.

[55] D. Zhang and J. Tsai, "Machine learning and software engineering," vol. 11, Feb. 2002, pp. 22–29, ISBN: 0-7695-1849-4. DOI: 10.1109/TAI.2002.1180784

[56] B. Kitchenham, T. Dybå, and M. Jorgensen, "Evidence-based software engineering," Jun. 2004, pp. 273–281, ISBN: 0-7695-2163-0. DOI: 10.1109/ICSE.2004.1317449

[57] V. Basili, F. Shull, and F. Lanubile, "Building knowledge through families of experiments," *IEEE Transactions on Software Engineering*, vol. 25, no. 4, pp. 456–473, 1999. DOI: 10.1109/32.799939

[58] P. E. Shrout and J. L. Rodgers, "Psychology, science, and knowledge construction: Broadening perspectives from the replication crisis," *Annual Review of Psychology*, vol. 69, no. Volume 69, 2018, pp. 487–510, 2018, ISSN: 1545-2085. DOI: https://doi.org/10.1146/annurev-psych-122216-011845 [Online]. Available: https://www.annualreviews.org/content/journals/10.1146/annurev-psych-122216-011845

[59] S. E. Maxwell, M. Y.-K. Lau, and G. S. Howard, "Is psychology suffering from a replication crisis? what does "failure to replicate" really mean?" *The American psychologist*, vol. 70 6, pp. 487–98, 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:11690879

[60] V. Amrhein, D. Trafimow, and S. Greenland, "Inferential statistics as descriptive statistics: There is no replication crisis if we don't expect replication," *The American Statistician*, vol. 73, pp. 262–270, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:56130495

[61] M. Cruz, B. Bernárdez, A. Durán, J. A. Galindo, and A. Ruiz-Cortés, "Replication of studies in empirical software engineering: A systematic mapping study, from 2013 to 2018," *IEEE Access*, vol. 8, pp. 26 773–26 791, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:209073520

[62] F. J. Shull, J. C. Carver, S. Vegas, and N. Juristo, "The role of replications in empirical software engineering," *Empirical Software Engineering*, vol. 13, no. 2, pp. 211–218, Jan. 2008, ISSN: 1573-7616. DOI: 10.1007/s10664-008-9060-1 [Online]. Available: http://dx.doi.org/10.1007/s10664-008-9060-1

[63] N. Juristo and S. Vegas, "The role of non-exact replications in software engineering experiments," *Empirical Software Engineering*, vol. 16, no. 3, pp. 295–324, Aug. 2010, ISSN: 1573-7616. DOI: 10.1007/s10664-010-9141-9 [Online]. Available: http://dx.doi.org/10.1007/s10664-010-9141-9

[64] M. Shepperd, N. Ajienka, and S. Counsell, "The role and value of replication in empirical software engineering results," *Information and Software Technology*, vol. 99, pp. 120–132, 2018, ISSN: 0950-5849. DOI: https://doi.org/10.1016/j.infsof.2018.01.006 [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584917304305

[65] M. Shepperd, "Replication studies considered harmful," in *Proceedings of the 40th International Conference on Software Engineering: New Ideas and Emerging Results*, ser. ICSE-NIER '18, Gothenburg, Sweden: Association for Computing Machinery, 2018, pp. 73–76, ISBN: 9781450356626. DOI: 10.1145/3183399.3183423 [Online]. Available: https://doi.org/10.1145/3183399.3183423

[66] J. Siegmund, N. Siegmund, and S. Apel, "Views on internal and external validity in empirical software engineering," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 1, 2015, pp. 9–19. DOI: 10.1109/ICSE.2015.24

[67] S. Vegas, N. Juristo, A. Moreno, M. Solari, and P. Letelier, "Analysis of the influence of communication between researchers on experiment replication," in *Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering*, ser. ISESE '06, Rio de Janeiro, Brazil: Association for Computing Machinery, 2006, pp. 28–37, ISBN: 1595932186. DOI: 10.1145/1159733.1159741 [Online]. Available: https://doi.org/10.1145/1159733.1159741

[68] O. S. Gómez, N. Juristo, and S. Vegas, "Understanding replication of experiments in software engineering: A classification," *Information and Software Technology*, vol. 56, no. 8, pp. 1033–1048, 2014, ISSN: 0950-5849. DOI: https://doi.org/10.1016/j.infsof.2014.04.004 [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584914000858

[69] S. Abualhaija et al., "Replication in requirements engineering: The nlp for re case," *ACM Trans. Softw. Eng. Methodol.*, vol. 33, no. 6, Jun. 2024, ISSN: 1049-331X. DOI: 10.1145/3658669 [Online]. Available: https://doi.org/10.1145/3658669

[70] M. J. Brandt et al., "The replication recipe: What makes for a convincing replication?" *Journal of Experimental Social Psychology*, vol. 50, pp. 217–224, 2014, ISSN: 0022-1031. DOI: https://doi.org/10.1016/j.jesp.2013.10.005 [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0022103113001819

[71] A. Fan et al., "Large language models for software engineering: Survey and open problems," in *2023 IEEE/ACM International Conference on Software Engineering: Future of Software Engineering (ICSE-FoSE)*, 2023, pp. 31–53. DOI: 10.1109/ICSE-FoSE59343.2023.00008

[72] L. Belzner, T. Gabor, and M. Wirsing, "Large language model assisted software engineering: Prospects, challenges, and a case study," in *Bridging the Gap Between AI and Reality*, B. Steffen, Ed., Cham: Springer Nature Switzerland, 2024, pp. 355–374, ISBN: 978-3-031-46002-9.

[73] Z. Zheng et al., "Towards an understanding of large language models in software engineering tasks," *Empirical Software Engineering*, vol. 30, no. 2, Dec. 2024, ISSN: 1573-7616. DOI: 10.1007/s10664-024-10602-0 [Online]. Available: http://dx.doi.org/10.1007/s10664-024-10602-0

[74] I. Ozkaya, "Application of large language models to software engineering tasks: Opportunities, risks, and implications," *IEEE Software*, vol. 40, no. 3, pp. 4–8, 2023. DOI: 10.1109/MS.2023.3248401

[75] J. Wang, Y. Huang, C. Chen, Z. Liu, S. Wang, and Q. Wang, *Software testing with large language models: Survey, landscape, and vision*, 2024. arXiv: 2307.07221 [cs.SE]. [Online]. Available: https://arxiv.org/abs/2307.07221

[76] S. Baltes et al., *Guidelines for empirical studies in software engineering involving large language models*, 2025. arXiv: 2508.15503 [cs.SE]. [Online]. Available: https://arxiv.org/abs/2508.15503

[77] C. Chen et al., "Clinicalbench: Can llms beat traditional ml models in clinical prediction?" *ArXiv*, vol. abs/2411.06469, 2024. [Online]. Available: https://api.semanticscholar.org/CorpusID:273963111

[78] S. Vajjala and S. Shimangaud, *Text classification in the llm era – where do we stand?* 2025. arXiv: 2502.11830 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2502.11830

[79] S. Ekin, "Prompt engineering for chatgpt: A quick guide to techniques, tips, and best practices," 2023. DOI: 10.36227/techrxiv.22683919.v2

[80] M. Sclar, Y. Choi, Y. Tsvetkov, and A. Suhr, *Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting*, 2024. arXiv: 2310.11324 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2310.11324

[81] M. Mizrahi, G. Kaplan, D. Malkin, R. Dror, D. Shahaf, and G. Stanovsky, *State of what art? a call for multi-prompt llm evaluation*, 2024. arXiv: 2401.00595 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2401.00595

[82] F. Maia Polo et al., *Efficient multi-prompt evaluation of llms*, May 2024. DOI: 10.48550/arXiv.2405.17202

[83] Q. Ye, M. Axmed, R. Pryzant, and F. Khani, *Prompt engineering a prompt engineer*, 2024. arXiv: 2311.05661 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2311.05661

[84] M. A. Zadenoori, L. Zhao, W. Alhoshan, and A. Ferrari, "Automatic prompt engineering: The case of requirements classification," in *International Working Conference on Requirements Engineering: Foundation for Software Quality*, Springer, 2025, pp. 217–225.

[85] Y. Liu et al., *Trustworthy llms: A survey and guideline for evaluating large language models' alignment*, 2024. arXiv: 2308.05374 [cs.AI]. [Online]. Available: https://arxiv.org/abs/2308.05374

[86] Z. R. Tam, C.-K. Wu, Y.-L. Tsai, C.-Y. Lin, H.-y. Lee, and Y.-N. Chen, "Let me speak freely? a study on the impact of format restrictions on large language model performance.," in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, F. Dernoncourt, D. Preoţiuc-Pietro, and A. Shimorina, Eds., Miami, Florida, US: Association for Computational Linguistics, Nov. 2024, pp. 1218–1236. DOI: 10.18653/v1/2024.emnlp-industry.91 [Online]. Available: https://aclanthology.org/2024.emnlp-industry.91/

[87] D. X. Long et al., *Llms are biased towards output formats! systematically evaluating and mitigating output format bias of llms*, 2025. arXiv: 2408.08656 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2408.08656

[88] D. Li, Y. Zhao, Z. Wang, C. Jung, and Z. Zhang, "Large language model-driven structured output: A comprehensive benchmark and spatial data generation framework," *ISPRS International Journal of Geo-Information*, vol. 13, no. 11, 2024, ISSN: 2220-9964. DOI: 10.3390/ijgi13110405 [Online]. Available: https://www.mdpi.com/2220-9964/13/11/405

[89] C. Xia et al., "FOFO: A benchmark to evaluate LLMs' format-following capability," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, L.-W. Ku, A. Martins, and V. Srikumar, Eds., Bangkok, Thailand: Association for Computational Linguistics, Aug. 2024, pp. 680–699. DOI: 10.18653/v1/2024.acl-long.40 [Online]. Available: https://aclanthology.org/2024.acl-long.40/

[90] Y. Chang et al., *A survey on evaluation of large language models*, 2023. arXiv: 2307.03109 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2307.03109

[91] C. X. Yu, J. James, C. Si Yu, D. David, C. B. Hoe, and P. Hui-Li Phyllis, "Can llms have a fever? investigating the effects of temperature on llm security," in *Proceedings of the International Conference on Industrial Engineering and Operations Management*, ser. South American '24, IEOM Society International, May 2024. DOI: 10.46254/sa05.20240024 [Online]. Available: http://dx.doi.org/10.46254/SA05.20240024

[92] J. Huang and C. X. Ling, "Using auc and accuracy in evaluating learning algorithms," *IEEE Transactions on knowledge and Data Engineering*, vol. 17, no. 3, pp. 299–310, 2005.

[93] C. X. Ling, J. Huang, and H. Zhang, "Auc: A better measure than accuracy in comparing learning algorithms," in *Conference of the canadian society for computational studies of intelligence*, Springer, 2003, pp. 329–341.

[94] F. S. Nahm, "Receiver operating characteristic curve: Overview and practical use for clinicians," *Korean Journal of Anesthesiology*, vol. 75, no. 1, pp. 25–36, Feb. 2022, ISSN: 2005-7563. DOI: 10.4097/kja.21209 [Online]. Available: http://dx.doi.org/10.4097/kja.21209

[95] J. Nixon, M. W. Dusenberry, L. Zhang, G. Jerfel, and D. Tran, "Measuring calibration in deep learning.," in *CVPR workshops*, vol. 2, 2019.

[96] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *International conference on machine learning*, PMLR, 2017, pp. 1321–1330.

[97] Z. Ma, A. R. Chen, D. J. Kim, T.-H. Chen, and S. Wang, "Llmparser: An exploratory study on using large language models for log parsing," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24, Lisbon, Portugal: Association for Computing Machinery, 2024, ISBN: 9798400702174. DOI: 10.1145/3597503.3639150 [Online]. Available: https://doi.org/10.1145/3597503.3639150

[98] M. Rinard, "Software engineering research in a world with generative artificial intelligence," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24, Lisbon, Portugal: Association for Computing Machinery, 2024, ISBN: 9798400702174. DOI: 10.1145/3597503.3649399 [Online]. Available: https://doi.org/10.1145/3597503.3649399

[99] Y. Peng, S. Gao, C. Gao, Y. Huo, and M. Lyu, "Domain knowledge matters: Improving prompts with fix templates for repairing python type errors," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24, Lisbon, Portugal: Association for Computing Machinery, 2024, ISBN: 9798400702174. DOI: 10.1145/3597503.3608132 [Online]. Available: https://doi.org/10.1145/3597503.3608132

[100] J. Xu et al., "Unilog: Automatic logging via llm and in-context learning," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24, Lisbon, Portugal: Association for Computing Machinery, 2024, ISBN: 9798400702174. DOI: 10.1145/3597503.3623326 [Online]. Available: https://doi.org/10.1145/3597503.3623326

[101] Y. Huang et al., "Crashtranslator: Automatically reproducing mobile application crashes directly from stack trace," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24, Lisbon, Portugal: Association for Computing Machinery, 2024, ISBN: 9798400702174. DOI: 10.1145/3597503.3623298 [Online]. Available: https://doi.org/10.1145/3597503.3623298

[102] Q. Guo et al., "Exploring the potential of chatgpt in automated code refinement: An empirical study," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24, Lisbon, Portugal: Association for Computing Machinery, 2024, ISBN: 9798400702174. DOI: 10.1145/3597503.3623306 [Online]. Available: https://doi.org/10.1145/3597503.3623306

[103] H. Yu et al., "Codereval: A benchmark of pragmatic code generation with generative pre-trained models," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24, Lisbon, Portugal: Association for Computing Machinery, 2024, ISBN: 9798400702174. DOI: 10.1145/3597503.3623316 [Online]. Available: https://doi.org/10.1145/3597503.3623316

[104] M. Geng et al., "Large language models are few-shot summarizers: Multi-intent comment generation via in-context learning," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24, Lisbon, Portugal: Association for Computing Machinery, 2024, ISBN: 9798400702174. DOI: 10.1145/3597503.3608134 [Online]. Available: https://doi.org/10.1145/3597503.3608134

[105] S. Feng and C. Chen, "Prompting is all you need: Automated android bug replay with large language models," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24, Lisbon, Portugal: Association for Computing Machinery, 2024, ISBN: 9798400702174. DOI: 10.1145/3597503.3608137 [Online]. Available: https://doi.org/10.1145/3597503.3608137

[106] Y. Deng, C. S. Xia, C. Yang, S. D. Zhang, S. Yang, and L. Zhang, "Large language models are edge-case generators: Crafting unusual programs for fuzzing deep learning libraries," in *Proceedings of the*

IEEE/ACM 46th International Conference on Software Engineering, ser. ICSE '24, Lisbon, Portugal: Association for Computing Machinery, 2024, ISBN: 9798400702174. DOI: 10.1145/3597503.3623343 [Online]. Available: https://doi.org/10.1145/3597503.3623343

[107] J. Chen, X. Hu, Z. Li, C. Gao, X. Xia, and D. Lo, "Code search is all you need? improving code suggestions with code search," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24, Lisbon, Portugal: Association for Computing Machinery, 2024, ISBN: 9798400702174. DOI: 10.1145/3597503.3639085 [Online]. Available: https://doi.org/10.1145/3597503.3639085

[108] Z. Sun, X. Du, F. Song, S. Wang, and L. Li, "When neural code completion models size up the situation: Attaining cheaper and faster completion through dynamic model inference," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24, Lisbon, Portugal: Association for Computing Machinery, 2024, ISBN: 9798400702174. DOI: 10.1145/3597503.3639120 [Online]. Available: https://doi.org/10.1145/3597503.3639120

[109] A. Al-Kaswan, M. Izadi, and A. van Deursen, "Traces of memorisation in large language models for code," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24, Lisbon, Portugal: Association for Computing Machinery, 2024, ISBN: 9798400702174. DOI: 10.1145/3597503.3639133 [Online]. Available: https://doi.org/10.1145/3597503.3639133

[110] M. Izadi, J. Katzy, T. Van Dam, M. Otten, R. M. Popescu, and A. Van Deursen, "Language models for code completion: A practical evaluation," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24, Lisbon, Portugal: Association for Computing Machinery, 2024, ISBN: 9798400702174. DOI: 10.1145/3597503.3639138 [Online]. Available: https://doi.org/10.1145/3597503.3639138

[111] X. Du et al., "Evaluating large language models in class-level code generation," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24, Lisbon, Portugal: Association for Computing Machinery, 2024, ISBN: 9798400702174. DOI: 10.1145/3597503.3639219 [Online]. Available: https://doi.org/10.1145/3597503.3639219

[112] R. Pan et al., "Lost in translation: A study of bugs introduced by large language models while translating code," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24, Lisbon, Portugal: Association for Computing Machinery, 2024, ISBN: 9798400702174. DOI: 10.1145/3597503.3639226 [Online]. Available: https://doi.org/10.1145/3597503.3639226

[113] Y. W. Chow, L. Di Grazia, and M. Pradel, "Pyty: Repairing static type errors in python," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24, Lisbon, Portugal: Association for Computing Machinery, 2024, ISBN: 9798400702174. DOI: 10.1145/3597503.3639184 [Online]. Available: https://doi.org/10.1145/3597503.3639184

[114] Y. Jiang et al., "Xpert: Empowering incident management with query recommendations via large language models," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24, Lisbon, Portugal: Association for Computing Machinery, 2024, ISBN: 9798400702174. DOI: 10.1145/3597503.3639081 [Online]. Available: https://doi.org/10.1145/3597503.3639081

[115] Y. Su et al., "Enhancing exploratory testing by large language model and knowledge graph," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24, Lisbon, Portugal: Association for Computing Machinery, 2024, ISBN: 9798400702174. DOI: 10.1145/3597503.3639157 [Online]. Available: https://doi.org/10.1145/3597503.3639157

[116] Z. Liu et al., "Make llm a testing expert: Bringing human-like interaction to mobile gui testing via functionality-aware decisions," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24, Lisbon, Portugal: Association for Computing Machinery, 2024, ISBN: 9798400702174. DOI: 10.1145/3597503.3639180 [Online]. Available: https://doi.org/10.1145/3597503.3639180

[117] C. S. Xia, M. Paltenghi, J. Le Tian, M. Pradel, and L. Zhang, "Fuzz4all: Universal fuzzing with large language models," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24, Lisbon, Portugal: Association for Computing Machinery, 2024, ISBN: 9798400702174. DOI: 10.1145/3597503.3639121 [Online]. Available: https://doi.org/10.1145/3597503.3639121

[118] Z. Liu et al., "Testing the limits: Unusual text inputs generation for mobile app crash detection with large language model," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24, Lisbon, Portugal: Association for Computing Machinery, 2024, ISBN: 9798400702174. DOI: 10.1145/3597503.3639118 [Online]. Available: https://doi.org/10.1145/3597503.3639118

[119] J. Fu, J. Liang, Z. Wu, and Y. Jiang, "Sedar: Obtaining high-quality seeds for dbms fuzzing via cross-dbms sql transfer," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24, Lisbon, Portugal: Association for Computing Machinery, 2024, ISBN: 9798400702174. DOI: 10.1145/3597503.3639210 [Online]. Available: https://doi.org/10.1145/3597503.3639210

[120] Y. Nong et al., "Vgx: Large-scale sample generation for boosting learning-based software vulnerability analyses," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24, Lisbon, Portugal: Association for Computing Machinery, 2024, ISBN: 9798400702174. DOI: 10.1145/3597503.3639116 [Online]. Available: https://doi.org/10.1145/3597503.3639116

[121] M. M. Imran, P. Chatterjee, and K. Damevski, "Uncovering the causes of emotions in software developer communication using zero-shot llms," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24, Lisbon, Portugal: Association for Computing Machinery, 2024, ISBN: 9798400702174. DOI: 10.1145/3597503.3639223 [Online]. Available: https://doi.org/10.1145/3597503.3639223

[122] J. Xu, R. Yang, Y. Huo, C. Zhang, and P. He, "Divlog: Log parsing with prompt enhanced in-context learning," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24, Lisbon, Portugal: Association for Computing Machinery, 2024, ISBN: 9798400702174. DOI: 10.1145/3597503.3639155 [Online]. Available: https://doi.org/10.1145/3597503.3639155

[123] T. Ahmed, K. S. Pai, P. Devanbu, and E. Barr, "Automatic semantic augmentation of language model prompts (for code summarization)," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24, Lisbon, Portugal: Association for Computing Machinery, 2024, ISBN: 9798400702174. DOI: 10.1145/3597503.3639183 [Online]. Available: https://doi.org/10.1145/3597503.3639183

[124] Y. Yang, X. Hu, X. Xia, D. Lo, and X. Yang, "Streamlining java programming: Uncovering well-formed idioms with idiomine," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24, Lisbon, Portugal: Association for Computing Machinery, 2024, ISBN: 9798400702174. DOI: 10.1145/3597503.3639135 [Online]. Available: https://doi.org/10.1145/3597503.3639135

[125] M. Toslali et al., "Agrabot: Accelerating third-party security risk management in enterprise setting through generative ai," in *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*, ser. FSE 2024, Porto de Galinhas, Brazil: Association for Com-

puting Machinery, 2024, pp. 74–79, ISBN: 9798400706585. DOI: 10.1145/3663529.3663829 [Online]. Available: https://doi.org/10.1145/3663529.3663829

[126] O. Dunay et al., "Multi-line ai-assisted code authoring," in *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*, ser. FSE 2024, Porto de Galinhas, Brazil: Association for Computing Machinery, 2024, pp. 150–160, ISBN: 9798400706585. DOI: 10.1145/3663529.3663836 [Online]. Available: https://doi.org/10.1145/3663529.3663836

[127] N. Alshahwan et al., "Automated unit test improvement using large language models at meta," in *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*, ser. FSE 2024, Porto de Galinhas, Brazil: Association for Computing Machinery, 2024, pp. 185–196, ISBN: 9798400706585. DOI: 10.1145/3663529.3663839 [Online]. Available: https://doi.org/10.1145/3663529.3663839

[128] D. Roy et al., "Exploring llm-based agents for root cause analysis," in *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*, ser. FSE 2024, Porto de Galinhas, Brazil: Association for Computing Machinery, 2024, pp. 208–219, ISBN: 9798400706585. DOI: 10.1145/3663529.3663841 [Online]. Available: https://doi.org/10.1145/3663529.3663841

[129] S. Wu et al., "Combating missed recalls in e-commerce search: A cot-prompting testing approach," in *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*, ser. FSE 2024, Porto de Galinhas, Brazil: Association for Computing Machinery, 2024, pp. 220–231, ISBN: 9798400706585. DOI: 10.1145/3663529.3663842 [Online]. Available: https://doi.org/10.1145/3663529.3663842

[130] X. Zhang et al., "Automated root causing of cloud incidents using in-context learning with gpt-4," in *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*, ser. FSE 2024, Porto de Galinhas, Brazil: Association for Computing Machinery, 2024, pp. 266–277, ISBN: 9798400706585. DOI: 10.1145/3663529.3663846 [Online]. Available: https://doi.org/10.1145/3663529.3663846

[131] K. Sarda, Z. Namrud, M. Litoiu, L. Shwartz, and I. Watts, "Leveraging large language models for the auto-remediation of microservice applications: An experimental study," in *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*, ser. FSE 2024, Porto de Galinhas, Brazil: Association for Computing Machinery, 2024, pp. 358–369, ISBN: 9798400706585. DOI: 10.1145/3663529.3663855 [Online]. Available: https://doi.org/10.1145/3663529.3663855

[132] D. Goel et al., "X-lifecycle learning for cloud incident management using llms," in *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*, ser. FSE 2024, Porto de Galinhas, Brazil: Association for Computing Machinery, 2024, pp. 417–428, ISBN: 9798400706585. DOI: 10.1145/3663529.3663861 [Online]. Available: https://doi.org/10.1145/3663529.3663861

[133] Y. Chen, Z. Hu, C. Zhi, J. Han, S. Deng, and J. Yin, "Chatunitest: A framework for llm-based test generation," in *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*, ser. FSE 2024, Porto de Galinhas, Brazil: Association for Computing Machinery, 2024, pp. 572–576, ISBN: 9798400706585. DOI: 10.1145/3663529.3663801 [Online]. Available: https://doi.org/10.1145/3663529.3663801

[134] H. Liu, Z. Jia, H. Zhou, H. Zhou, and S. Li, "Go the extra mile: Fixing propagated error-handling bugs," in *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*, ser. FSE 2024, Porto de Galinhas, Brazil: Association for Computing Machinery, 2024, pp. 661–662, ISBN: 9798400706585. DOI: 10.1145/3663529.3663868 [Online]. Available: https://doi.org/10.1145/3663529.3663868

[135] S. B. Chavan and S. Mondal, “Do large language models recognize python identifier swaps in their generated code?” In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*, ser. FSE 2024, Porto de Galinhas, Brazil: Association for Computing Machinery, 2024, pp. 663–664, ISBN: 9798400706585. DOI: 10.1145/3663529.3663869 [Online]. Available: https://doi.org/10.1145/3663529.3663869

[136] H. Patel, K. A. Shah, and S. Mondal, “Do large language models generate similar codes from mutated prompts? a case study of gemini pro,” in *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*, ser. FSE 2024, Porto de Galinhas, Brazil: Association for Computing Machinery, 2024, pp. 671–672, ISBN: 9798400706585. DOI: 10.1145/3663529.3663873 [Online]. Available: https://doi.org/10.1145/3663529.3663873

[137] D. Souza, “Comparing gemini pro and gpt-3.5 in algorithmic problems,” in *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*, ser. FSE 2024, Porto de Galinhas, Brazil: Association for Computing Machinery, 2024, pp. 698–700, ISBN: 9798400706585. DOI: 10.1145/3663529.3664463 [Online]. Available: https://doi.org/10.1145/3663529.3664463

[138] M. Schafer, S. Nadi, A. Eghbali, and F. Tip, “ An Empirical Evaluation of Using Large Language Models for Automated Unit Test Generation,” *IEEE Transactions on Software Engineering*, vol. 50, no. 01, pp. 85–105, Jan. 2024, ISSN: 1939-3520. DOI: 10.1109/TSE.2023.3334955 [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/TSE.2023.3334955

[139] R. Tufano, O. Dabic, A. Mastropaolo, M. Ciniselli, and G. Bavota, “ Code Review Automation: Strengths and Weaknesses of the State of the Art,” *IEEE Transactions on Software Engineering*, vol. 50, no. 02, pp. 338–353, Feb. 2024, ISSN: 1939-3520. DOI: 10.1109/TSE.2023.3348172 [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/TSE.2023.3348172

[140] Y. Zhang et al., “ Automatic Commit Message Generation: A Critical Review and Directions for Future Work,” *IEEE Transactions on Software Engineering*, vol. 50, no. 04, pp. 816–835, Apr. 2024, ISSN: 1939-3520. DOI: 10.1109/TSE.2024.3364675 [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/TSE.2024.3364675

[141] Y. Tang, Z. Liu, Z. Zhou, and X. Luo, “ ChatGPT vs SBST: A Comparative Assessment of Unit Test Suite Generation,” *IEEE Transactions on Software Engineering*, vol. 50, no. 06, pp. 1340–1359, Jun. 2024, ISSN: 1939-3520. DOI: 10.1109/TSE.2024.3382365 [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/TSE.2024.3382365

[142] Z. Liu, Y. Tang, X. Luo, Y. Zhou, and L. F. Zhang, “ No Need to Lift a Finger Anymore? Assessing the Quality of Code Generation by ChatGPT,” *IEEE Transactions on Software Engineering*, vol. 50, no. 06, pp. 1548–1584, Jun. 2024, ISSN: 1939-3520. DOI: 10.1109/TSE.2024.3392499 [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/TSE.2024.3392499

[143] C. Liu, P. Cetin, Y. Patodia, B. Ray, S. Chakraborty, and Y. Ding, “ Automated Code Editing With Search-Generate-Modify,” *IEEE Transactions on Software Engineering*, vol. 50, no. 07, pp. 1675–1686, Jul. 2024, ISSN: 1939-3520. DOI: 10.1109/TSE.2024.3376387 [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/TSE.2024.3376387

[144] H. Tu, Z. Zhou, H. Jiang, I. N. B. Yusuf, Y. Li, and L. Jiang, “ Isolating Compiler Bugs by Generating Effective Witness Programs With Large Language Models,” *IEEE Transactions on Software Engineering*, vol. 50, no. 07, pp. 1768–1788, Jul. 2024, ISSN: 1939-3520. DOI: 10.1109/TSE.2024.3397822 [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/TSE.2024.3397822

[145] C. Fang et al., " Esale: Enhancing Code-Summary Alignment Learning for Source Code Summarization," *IEEE Transactions on Software Engineering*, vol. 50, no. 08, pp. 2077–2095, Aug. 2024, ISSN: 1939-3520. DOI: 10.1109/TSE.2024.3422274 [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/TSE.2024.3422274

[146] S. Fakhoury, A. Naik, G. Sakkas, S. Chakraborty, and S. K. Lahiri, " LLM-Based Test-Driven Interactive Code Generation: User Study and Empirical Evaluation," *IEEE Transactions on Software Engineering*, vol. 50, no. 09, pp. 2254–2268, Sep. 2024, ISSN: 1939-3520. DOI: 10.1109/TSE.2024.3428972 [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/TSE.2024.3428972

[147] J. Shin, H. Hemmati, M. Wei, and S. Wang, " Assessing Evaluation Metrics for Neural Test Oracle Generation," *IEEE Transactions on Software Engineering*, vol. 50, no. 09, pp. 2337–2349, Sep. 2024, ISSN: 1939-3520. DOI: 10.1109/TSE.2024.3433463 [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/TSE.2024.3433463

[148] G. Yang, Y. Zhou, X. Chen, X. Zhang, T. Y. Zhuo, and T. Chen, " Chain-of-Thought in Neural Code Generation: From and for Lightweight Language Models," *IEEE Transactions on Software Engineering*, vol. 50, no. 09, pp. 2437–2457, Sep. 2024, ISSN: 1939-3520. DOI: 10.1109/TSE.2024.3440503 [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/TSE.2024.3440503

[149] Z. Zhou, M. Li, H. Yu, G. Fan, P. Yang, and Z. Huang, " Learning to Generate Structured Code Summaries From Hybrid Code Context," *IEEE Transactions on Software Engineering*, vol. 50, no. 10, pp. 2512–2528, Oct. 2024, ISSN: 1939-3520. DOI: 10.1109/TSE.2024.3439562 [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/TSE.2024.3439562

[150] S. Kang, J. Yoon, N. Askarbekkyzy, and S. Yoo, " Evaluating Diverse Large Language Models for Automatic and General Bug Reproduction," *IEEE Transactions on Software Engineering*, vol. 50, no. 10, pp. 2677–2694, Oct. 2024, ISSN: 1939-3520. DOI: 10.1109/TSE.2024.3450837 [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/TSE.2024.3450837

[151] H. Wang, Z. Gao, X. Hu, D. Lo, J. Grundy, and X. Wang, " Just-In-Time TODO-Missed Commits Detection," *IEEE Transactions on Software Engineering*, vol. 50, no. 11, pp. 2732–2752, Nov. 2024, ISSN: 1939-3520. DOI: 10.1109/TSE.2024.3405005 [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/TSE.2024.3405005

[152] Y. Li et al., " Exploring the Effectiveness of LLMs in Automated Logging Statement Generation: An Empirical Study," *IEEE Transactions on Software Engineering*, vol. 50, no. 12, pp. 3188–3207, Dec. 2024, ISSN: 1939-3520. DOI: 10.1109/TSE.2024.3475375 [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/TSE.2024.3475375

[153] P. Xue et al., " Automated Commit Message Generation With Large Language Models: An Empirical Study and Beyond," *IEEE Transactions on Software Engineering*, vol. 50, no. 12, pp. 3208–3224, Dec. 2024, ISSN: 1939-3520. DOI: 10.1109/TSE.2024.3478317 [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/TSE.2024.3478317

[154] D. Liao et al., " $\mathbf{A^3}$A3-CodGen: A Repository-Level Code Generation Framework for Code Reuse With Local-Aware, Global-Aware, and Third-Party-Library-Aware," *IEEE Transactions on Software Engineering*, vol. 50, no. 12, pp. 3369–3384, Dec. 2024, ISSN: 1939-3520. DOI: 10.1109/TSE.2024.3486195 [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/TSE.2024.3486195

[155] B. Steenhoek, H. Gao, and W. Le, "Dataflow analysis-inspired deep learning for efficient vulnerability detection," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24, Lisbon, Portugal: Association for Computing Machinery, 2024, ISBN: 9798400702174.

DOI: 10.1145/3597503.3623345 [Online]. Available: https://doi.org/10.1145/3597503.3623345

[156] L. Ma et al., "Knowlog: Knowledge enhanced pre-trained language model for log understanding," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24, Lisbon, Portugal: Association for Computing Machinery, 2024, ISBN: 9798400702174. DOI: 10.1145/3597503.3623304 [Online]. Available: https://doi.org/10.1145/3597503.3623304

[157] Z. Zhou, C. Sha, and X. Peng, "On calibration of pre-trained code models," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24, Lisbon, Portugal: Association for Computing Machinery, 2024, ISBN: 9798400702174. DOI: 10.1145/3597503.3639126 [Online]. Available: https://doi.org/10.1145/3597503.3639126

[158] W. Wang, Y. Li, A. Li, J. Zhang, W. Ma, and Y. Liu, "An empirical study on noisy label learning for program understanding," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24, Lisbon, Portugal: Association for Computing Machinery, 2024, ISBN: 9798400702174. DOI: 10.1145/3597503.3639217 [Online]. Available: https://doi.org/10.1145/3597503.3639217

[159] Y. Sun et al., "Gptscan: Detecting logic vulnerabilities in smart contracts by combining gpt with program analysis," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24, Lisbon, Portugal: Association for Computing Machinery, 2024, ISBN: 9798400702174. DOI: 10.1145/3597503.3639117 [Online]. Available: https://doi.org/10.1145/3597503.3639117

[160] M. H. Tanzil, J. Y. Khan, and G. Uddin, "Chatgpt incorrectness detection in software reviews," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24, Lisbon, Portugal: Association for Computing Machinery, 2024, ISBN: 9798400702174. DOI: 10.1145/3597503.3639194 [Online]. Available: https://doi.org/10.1145/3597503.3639194

[161] J. Sun, J. Chen, Z. Xing, Q. Lu, X. Xu, and L. Zhu, "Where is it? tracing the vulnerability-relevant files from vulnerability reports," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24, Lisbon, Portugal: Association for Computing Machinery, 2024, ISBN: 9798400702174. DOI: 10.1145/3597503.3639202 [Online]. Available: https://doi.org/10.1145/3597503.3639202

[162] Y. Wang, J. A. H. López, U. Nilsson, and D. Varró, "Using run-time information to enhance static analysis of machine learning code in notebooks," in *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*, ser. FSE 2024, Porto de Galinhas, Brazil: Association for Computing Machinery, 2024, pp. 497–501, ISBN: 9798400706585. DOI: 10.1145/3663529.3663785 [Online]. Available: https://doi.org/10.1145/3663529.3663785

[163] A. Hora, "Predicting test results without execution," in *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*, ser. FSE 2024, Porto de Galinhas, Brazil: Association for Computing Machinery, 2024, pp. 542–546, ISBN: 9798400706585. DOI: 10.1145/3663529.3663794 [Online]. Available: https://doi.org/10.1145/3663529.3663794

[164] Y. Shen and T. Breaux, " Stakeholder Preference Extraction From Scenarios," *IEEE Transactions on Software Engineering*, vol. 50, no. 01, pp. 69–84, Jan. 2024, ISSN: 1939-3520. DOI: 10.1109/TSE.2023.3333265 [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/TSE.2023.3333265

[165] Q. Zhang et al., " APPT: Boosting Automated Patch Correctness Prediction via Fine-Tuning Pre-Trained Models," *IEEE Transactions on Software Engineering*, vol. 50, no. 03, pp. 474–494, Mar.

2024, ISSN: 1939-3520. DOI: 10.1109/TSE.2024.3354969 [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/TSE.2024.3354969

[166] L. Cui et al., " API2Vec++: Boosting API Sequence Representation for Malware Detection and Classification," *IEEE Transactions on Software Engineering*, vol. 50, no. 08, pp. 2142–2162, Aug. 2024, ISSN: 1939-3520. DOI: 10.1109/TSE.2024.3422990 [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/TSE.2024.3422990

[167] W. Sun, Z. Guo, M. Yan, Z. Liu, Y. Lei, and H. Zhang, " Method-Level Test-to-Code Traceability Link Construction by Semantic Correlation Learning," *IEEE Transactions on Software Engineering*, vol. 50, no. 10, pp. 2656–2676, Oct. 2024, ISSN: 1939-3520. DOI: 10.1109/TSE.2024.3449917 [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/TSE.2024.3449917

[168] X. Zhou et al., " Leveraging Large Language Model for Automatic Patch Correctness Assessment," *IEEE Transactions on Software Engineering*, vol. 50, no. 11, pp. 2865–2883, Nov. 2024, ISSN: 1939-3520. DOI: 10.1109/TSE.2024.3452252 [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/TSE.2024.3452252

[169] Y. Jiang, Y. Zhang, X. Su, C. Treude, and T. Wang, " StagedVulBERT: Multigranular Vulnerability Detection With a Novel Pretrained Code Model," *IEEE Transactions on Software Engineering*, vol. 50, no. 12, pp. 3454–3471, Dec. 2024, ISSN: 1939-3520. DOI: 10.1109/TSE.2024.3493245 [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/TSE.2024.3493245

[170] A. Z. H. Yang, C. Le Goues, R. Martins, and V. Hellendoorn, "Large language models for test-free fault localization," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24, Lisbon, Portugal: Association for Computing Machinery, 2024, ISBN: 9798400702174. DOI: 10.1145/3597503.3623342 [Online]. Available: https://doi.org/10.1145/3597503.3623342

[171] Y. Cai, A. Yadavally, A. Mishra, G. Montejo, and T. Nguyen, "Programming assistant for exception handling with codebert," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24, Lisbon, Portugal: Association for Computing Machinery, 2024, ISBN: 9798400702174. DOI: 10.1145/3597503.3639188 [Online]. Available: https://doi.org/10.1145/3597503.3639188

[172] D. Nam, A. Macvean, V. Hellendoorn, B. Vasilescu, and B. Myers, "Using an llm to help with code understanding," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24, Lisbon, Portugal: Association for Computing Machinery, 2024, ISBN: 9798400702174. DOI: 10.1145/3597503.3639187 [Online]. Available: https://doi.org/10.1145/3597503.3639187

[173] Z. Yu et al., "Monitorassistant: Simplifying cloud service monitoring via large language models," in *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*, ser. FSE 2024, Porto de Galinhas, Brazil: Association for Computing Machinery, 2024, pp. 38–49, ISBN: 9798400706585. DOI: 10.1145/3663529.3663826 [Online]. Available: https://doi.org/10.1145/3663529.3663826

[174] D. Pomian et al., "Em-assist: Safe automated extractmethod refactoring with llms," in *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*, ser. FSE 2024, Porto de Galinhas, Brazil: Association for Computing Machinery, 2024, pp. 582–586, ISBN: 9798400706585. DOI: 10.1145/3663529.3663803 [Online]. Available: https://doi.org/10.1145/3663529.3663803

[175] R. Pan, T. A. Ghaleb, and L. C. Briand, " LTM: Scalable and Black-Box Similarity-Based Test Suite Minimization Based on Language Models," *IEEE Transactions on Software Engineering*, vol. 50, no. 11, pp. 3053–3070, Nov. 2024, ISSN: 1939-3520. DOI: 10.1109/TSE.2024.3469582

[Online]. Available: https://doi.ieeecomputersociety.org/10.1109/TSE.2024.3469582

[176] S. Gao et al., "Learning in the wild: Towards leveraging unlabeled data for effectively tuning pre-trained code models," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24, Lisbon, Portugal: Association for Computing Machinery, 2024, ISBN: 9798400702174. DOI: 10.1145/3597503.3639216 [Online]. Available: https://doi.org/10.1145/3597503.3639216

[177] M. Nashaat and J. Miller, " Towards Efficient Fine-Tuning of Language Models With Organizational Data for Automated Software Review," *IEEE Transactions on Software Engineering*, vol. 50, no. 09, pp. 2240–2253, Sep. 2024, ISSN: 1939-3520. DOI: 10.1109/TSE.2024.3428324 [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/TSE.2024.3428324

[178] S. Fatima, H. Hemmati, and L. C. Briand, " FlakyFix: Using Large Language Models for Predicting Flaky Test Fix Categories and Test Code Repair," *IEEE Transactions on Software Engineering*, vol. 50, no. 12, pp. 3146–3171, Dec. 2024, ISSN: 1939-3520. DOI: 10.1109/TSE.2024.3472476 [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/TSE.2024.3472476

[179] Y. Zhang et al., "Learning-based widget matching for migrating gui test cases," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24, Lisbon, Portugal: Association for Computing Machinery, 2024, ISBN: 9798400702174. DOI: 10.1145/3597503.3623322 [Online]. Available: https://doi.org/10.1145/3597503.3623322

Appendices

A Supplemental Mapping Study Results

For brevity, sources were excluded for some figures and tables and are included here for further reference. Table 6 shows the distribution of the application types of LLMs of 83 total papers (see Figure 2).

Table 6 References for the distribution of LLM application types.

Category	Number of studies	References
Generation	58	[97–154]
Classification	15	[155–169]
Recommendation	6	[170–175]
Generation & Classification	3	[176–178]
Classification & Recommendation	1	[179]

The following tables apply to the 18 papers included in the mapping study. Table 7 shows the number of studies belonging to each venue. Table 8 shows the number of papers using prompts for classification. See also Figure 3 and 4.

Table 7 References for distribution of venue in the mapping study.

Venue	Number of studies	References
TSE	8	[164–169, 177, 178]
FSE	2	[162, 163]
ICSE	8	[155, 156, 158–161, 176, 179]

Table 8 References for the distribution of prompt usage in the mapping study.

Prompt-based	Number of studies	References
No	10	[155, 156, 158, 161, 164–166, 176, 178, 179]
Yes	8	[159, 160, 162, 163, 167–169, 177]

For the main results of the mapping study, Table 9 shows the full references. Further, Table 10 shows the references for the papers that use a prompt-based approach to classification.

Table 9 References for the mapping study results.

Category	Total	References
Total mapping study papers	18	[155, 156, 158–169, 176–179]
Evaluation metrics		
Precision	17 (94.4%)	[155, 156, 158–167, 169, 176–179]
Recall	17 (94.4%)	[155, 156, 158–167, 169, 176–179]
Accuracy	11 (61.1%)	[156, 158, 160, 161, 163–166, 168, 169, 177]
F-Score	17 (94.4%)	[155, 156, 158–162, 164–169, 176–179]
AUC	3 (16.7%)	[161, 165, 168]
ROC plots	0 (0.0%)	
MCC	0 (0.0%)	
Calibration	0 (0.0%)	
Fairness	0 (0.0%)	
Robustness		
Data distribution & Adversarial Attacks	1 (5.5%)	[166]
Noisy data	1 (5.5%)	[158]
Metrics justification		
No	9 (50.0%)	[158–164, 166, 177]
Implicit	0 (0.0%)	
Previous work	5 (27.8%)	[165, 168, 169, 176, 179]
Yes	4 (22.2%)	[155, 156, 167, 178]
Confusion matrix	4 (22.2%)	[159, 163, 167, 168]
Evaluation over multiple data sets	5 (27.8%)	[159, 165, 167, 176, 179]
Type of baseline		
None	2 (11.1%)	[159, 163]
Own	3 (16.7%)	[160, 162, 178]
External	4 (22.2%)	[156, 165, 177, 179]
External and Own	9 (50.0%)	[155, 158, 161, 164, 166–169, 176]
Analysis of statistical significance	5 (27.8%)	[155, 167–169, 178]
Statistical significance justification		
No	4 (80.0%)	[167–169, 178]
Implicit	0 (0.0%)	
Previous work	1 (20.0%)	[155]
Yes	0 (0.0%)	

Table 10 References for the mapping study prompting results.

Category	Total	References
Total prompt-based classification papers	8	[159, 160, 162, 163, 167–169, 177]
Prompt approach		
Chain-of-verification	1 (12.5%)	[160]
Few-shot	2 (25.0%)	[169, 177]
Mimic-in-the-background	1 (12.5%)	[159]
Not reported	1 (12.5%)	[162]
Zero-shot	3 (37.5%)	[163, 167, 168]
Prompt pattern		
Not reported	2 (25.0%)	[162, 169]
Own	6 (75.0%)	[159, 160, 163, 167, 168, 177]
Prompt approach justification		
No	5 (62.5%)	[162, 163, 167–169]
Yes	3 (37.5%)	[159, 160, 177]
Prompt design justification		
No	4 (50.0%)	[160, 162, 167, 169]
Previous Work	1 (12.5%)	[163]
Yes	3 (37.5%)	[159, 168, 177]
Evaluation over multiple prompts		
No	8 (100.0%)	[159, 160, 162, 163, 167–169, 177]
Temperature		
No	7 (87.5%)	[160, 162, 163, 167–169, 177]
Yes	1 (12.5%)	[159]