**Freie Universität Berlin**

Masterarbeit am Institut für Informatik der Freien Universität Berlin

Arbeitsgruppe Software Engineering

# Large Language Models in Software Engineering: A Critical Review of Evaluation Strategies

Ali Bektas

Matrikelnummer: 5404368

alib10@zedat.fu-berlin.de

Betreuerin: Carina Haupt

Eingereicht bei: Prof. Dr. Lutz Prechelt

Zweitgutachter: Prof. Dr. Michael Felderer

Berlin, January 20, 2025

**Abstract**

The advent of Transformer-based Large Language Models (LLMs) has significantly influenced Software Engineering (SE), enabling advancements in tasks such as requirements engineering, bug detection, and API documentation alongside traditional code generation applications. In recent years, research in this domain has grown substantially, with numerous studies exploring how LLMs can be applied to diverse SE tasks. While several surveys address the use of LLMs in SE [20, 12, 89] and general evaluation strategies for LLMs [5, 16, 7], a detailed analysis of evaluation strategies specifically tailored to SE tasks remains lacking. Such an analysis is essential to ensure that evaluations accurately reflect the real-world applicability of LLM-based solutions in SE.

This thesis systematically examines the evaluation strategies employed in LLM-based SE research. Using a structured six-phase approach, a subset of 41 non-code-centric studies derived from the corpus identified by Hou et al. [20] was critically reviewed. The analysis found that reliability was moderate, reflecting adherence to baseline methodological rigor but hindered by challenges such as dataset generalizability and transparency. Relevance, on the other hand, was moderate to high, though evaluations varied in how well metrics aligned with task objectives. Researchers often faced trade-offs between rapid solution development and evaluation rigor, leading to persistent baseline-level limitations.

The study highlights key opportunities to improve evaluation practices. Collaboration between academia and industry is critical to addressing dataset limitations by developing methods to create representative datasets that reflect industrial workflows while respecting confidentiality. Task-specific best practices can be developed by synthesizing strengths from existing evaluation strategies, ensuring alignment with SE tasks. Additionally, incorporating LLM-specific traits, such as variability in probability distributions and the influence of temperature settings, can improve evaluation reliability and better reflect real-world performance.

By addressing these gaps and implementing the proposed improvements, this thesis provides actionable recommendations to enhance the reliability and relevance of LLM evaluation strategies in SE. These advancements are crucial for ensuring that LLM-based solutions deliver meaningful and impactful outcomes in both academic and industrial contexts.

**Zusammenfassung**

Die Entwicklung von auf Transformern basierenden großen Sprachmodellen (LLMs) hat die Softwareentwicklung (SE) maßgeblich beeinflusst und Fortschritte in Bereichen wie Anforderungsmanagement, Fehlererkennung und API-Dokumentation ermöglicht – zusätzlich zu den etablierten Anwendungen im Bereich der Code-Generierung. In den letzten Jahren hat die Forschung in diesem Bereich erheblich zugenommen, wobei zahlreiche Studien untersuchen, wie LLMs auf verschiedene Aufgaben der Softwareentwicklung angewendet werden können. Während einige Übersichtsarbeiten die Nutzung von LLMs in der SE [20, 12, 89] sowie bestehende Ansätze zur Evaluation von LLMs [5, 16, 7] zusammenfassen, fehlt bisher eine detaillierte Analyse der Evaluierungsstrategien, die speziell auf SE-Aufgaben zugeschnitten sind. Eine solche Analyse ist jedoch entscheidend,

um sicherzustellen, dass die Bewertung von LLM-basierten Lösungen deren Praxistauglichkeit in der SE realistisch widerspiegelt.

Diese Arbeit untersucht systematisch die in der LLM-basierten SE-Forschung eingesetzten Evaluierungsstrategien. Dabei wurde ein strukturierter Ansatz in sechs Phasen verfolgt, um 41 nicht-code-zentrierte Studien aus dem von Hou et al. [20] identifizierten Korpus kritisch zu analysieren. Die Ergebnisse zeigen, dass die Zuverlässigkeit der Evaluierungen moderat ist: Zwar wurde eine grundsätzliche methodische Sorgfalt eingehalten, doch beeinträchtigten Herausforderungen wie mangelnde Generalisierbarkeit und Transparenz der Datensätze die Ergebnisse. Die Relevanz der Evaluierungen wurde hingegen als moderat bis hoch eingeschätzt, wobei die Metriken in unterschiedlichem Maße an die spezifischen Aufgaben angepasst waren, für die sie die Qualität der Lösungen bewerten sollten. Häufig mussten Forschende Kompromisse zwischen der schnellen Entwicklung von Lösungen und einer rigorosen Evaluierung eingehen, was zu grundlegenden Einschränkungen bei den Baseline-Ansätzen führte.

Die Studie identifiziert zentrale Ansatzpunkte zur Verbesserung der Evaluierungspraxis. Eine stärkere Zusammenarbeit zwischen Wissenschaft und Industrie ist entscheidend, um Datenlücken zu schließen und Methoden zur Erstellung repräsentativer Datensätze zu entwickeln, die industrielle Workflows widerspiegeln, dabei jedoch Vertraulichkeitsanforderungen berücksichtigen. Durch die Synthese bewährter Ansätze können aufgabenspezifische Best Practices entwickelt werden, die besser mit den Zielen der SE-Aufgaben übereinstimmen. Darüber hinaus kann die Berücksichtigung von LLM-spezifischen Eigenschaften, wie der Variabilität in Wahrscheinlichkeitsverteilungen und dem Einfluss von Temperatureinstellungen, die Zuverlässigkeit der Evaluierungen verbessern und die realen Leistungsfähigkeiten präziser abbilden.

Durch die Identifizierung dieser Schwachstellen und die Umsetzung der vorgeschlagenen Verbesserungen bietet diese Arbeit praxisorientierte Empfehlungen, um die Zuverlässigkeit und Relevanz von LLM-Evaluierungsstrategien in der SE zu steigern. Diese Fortschritte sind von zentraler Bedeutung, um sicherzustellen, dass LLM-basierte Lösungen sowohl in der Forschung als auch in industriellen Anwendungen messbare und nachhaltige Ergebnisse liefern.

## Eidesstattliche Erklärung

Ich versichere hiermit an Eides Statt, dass diese Arbeit von niemand anderem als meiner Person verfasst worden ist. Alle verwendeten Hilfsmittel wie Berichte, Bücher, Internetseiten oder ähnliches sind im Literaturverzeichnis angegeben, Zitate aus fremden Arbeiten sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

January 20, 2025

Ali Bektas

# Contents

# List of Figures



# List of Tables

# 1 Introduction

The introduction of transformer-based models has significantly influenced the field of natural language processing (NLP), particularly through the use of attention mechanisms. These mechanisms allow models to process input sequences in parallel, greatly enhancing their ability to capture context and relationships within the text. This advancement has led to the widespread adoption of Large Language Models (LLMs) across various domains, including Software Engineering (SE).

As the application of LLMs in SE expands, there has been a notable increase in both academic research and industry adoption. The number of studies exploring the use of LLMs for SE tasks has grown significantly, as documented by Hou et al. [20]. Researchers are applying LLMs to a variety of SE tasks such as code generation, bug detection, comment generation, and test generation. Meanwhile, industry has also begun integrating LLMs into real-world SE workflows. For example, GitHub Copilot[1] provides LLM-powered code suggestions within integrated development environments (IDEs), facilitating pair programming and boosting developer productivity. Additionally, IDEs are embedding LLMs to support pair programming, with courses and tools being developed to help engineers leverage these capabilities[2]. Tools like IBM Engineering Requirements Quality Assistant[3] are also using LLMs to assist with requirement management, automating complex aspects of SE workflows.

While the potential of LLMs in SE is widely acknowledged, the evaluation of these models within SE tasks presents unique challenges. Ensuring that the evaluation methods used in research are reliable and relevant to real-world SE applications is critical for understanding how these models perform and for guiding future research. Existing surveys and studies primarily focus on various aspects of LLM usage in SE, from general applications to task-specific evaluations, but a comprehensive review of evaluation strategies tailored specifically for SE tasks is lacking. This gap makes it difficult to assess the reliability and relevance of current evaluation practices when applied to SE tasks. Since the quality of evaluation directly impacts the perceived effectiveness of LLM-based solutions, it is essential to critically examine the methods used to ensure they provide meaningful insights into the models' practical utility.

The goal of this thesis is to address this gap by systematically reviewing the evaluation strategies employed in research on LLM-based SE solutions. By assessing the reliability and relevance of these evaluations, this study aims to provide a clearer understanding of how well current methodologies align with the practical needs of SE tasks. Through a structured analysis of the literature, the research seeks to identify strengths and weaknesses in existing approaches, contributing to the development of more robust and effective evaluation strategies for LLM applications in SE.

The structure of this thesis reflects the systematic approach taken to analyze evaluation strategies for LLM-based Software Engineering tasks. Section 2 reviews the state of the art in LLMs (2.1) and their applications in Software Engineering, highlighting advancements, challenges, and opportunities (2.2-2.3). Section 3 presents related work and describes the research corpora used in this study, including the replication

[1] https://github.com/features/copilot

[2] https://www.deeplearning.ai/short-courses/pair-programming-llm/

[3] https://www.ibm.com/docs/en/erqa?topic=assistant-overview

package provided by Hou et al. [20]. Section 4 starts by introducing the six-phase methodological framework applied in this research (4.1), detailing the preparation of the review (4.2), the group-wise review process organized by task objectives (4.3.1), and the criteria used to assess reliability and relevance (4.2.2). Section 5 summarizes the findings, including descriptive overviews of the review organized by task objective groups (5.1) and critical analyses of strengths and limitations across dimensions such as task objectives, research focus, ML task types, and publication trends (5.2). It also examines reliability and relevance, identifying patterns and contrasts across evaluation practices (5.3). Section 6 synthesizes these findings to address the research questions on reliability (6.1), relevance (6.2), and approaches to addressing limitations in evaluation strategies (6.3). Finally, Section 7 provides a summary of the key contributions of this work, discusses its implications for the field, and outlines directions for future research.

## 1.1 Motivation

The increasing adoption of LLMs in SE offers significant potential for automating complex tasks such as code generation, bug detection, and test generation. However, despite their growing use, there is a critical gap in the way LLM-based solutions are evaluated in SE research. Current evaluation methods often rely on conventional metrics like accuracy, precision, and F1-score, which may not fully capture the complexities and real-world challenges posed by SE tasks. These standard metrics may overlook important dimensions such as interpretability, robustness, and contextual relevance, which are essential in SE applications [5, 16].

As suggested by recent studies and highlighted in this work, many existing research papers likely fall short in providing reliable or relevant evaluations of LLMs for SE tasks. This poses a risk of misleading results, overestimating the effectiveness of LLMs, or failing to address critical areas where these models underperform in practical applications. By focusing on a systematic review of evaluation strategies, this research aims to unveil these limitations and provide a structured critique of the current practices.

Uncovering these gaps is essential not only for academic progress but also for the broader SE community, where the reliability and relevance of LLM-based tools directly impact the quality and success of software development workflows. Improved evaluation methods will ensure that research results more accurately reflect the capabilities of LLMs in SE, guiding both future research efforts and practical industry applications. This study, therefore, aims to elevate the evaluation standards in SE research and contribute to the development of more rigorous and meaningful methodologies that align with real-world SE needs.

## 1.2 Research Questions

Motivated by the observation that evaluation strategies for LLM-based solutions in software engineering (SE) may face challenges in effectively assessing their real-world applicability, this thesis investigates the following research questions:

- **RQ1:** How reliable are the evaluation strategies used in LLM-based software engineering research?
- **RQ2:** How relevant are the evaluation strategies in reflecting real-world software engineering needs?
- **RQ3:** What are the key gaps and limitations in current evaluation strategies, and how can they be addressed?

These questions aim to critically examine existing evaluation practices, identify potential shortcomings, and provide structured insights to guide improvements in future research and practical applications.

# 2 State of the Art

## 2.1 Large Language Models

Natural Language Processing (NLP) is a domain within artificial intelligence that focuses on the interaction between computers and humans through natural language. The objective is to enable computers to understand, interpret, and generate human language to support tasks like translation, summarization, information retrieval, sentiment analysis, and more. Within this field, a language model is a computational tool that predicts the likelihood of a sequence of words or phrases, capturing the essence of language syntax and semantics based on vast amounts of training text data.

The evolution of language models has unfolded through distinct phases, each marked by significant technological milestones. In the 1950s and 1960s, the field was dominated by rule-based models, relying on manually crafted linguistic rules to process language. The 1980s and especially the 1990s witnessed a pivotal shift to statistical models that used large text corpora to infer language patterns, enhancing the adaptability and scalability of language processing.

The late 1990s and early 2000s introduced neural network-based models, notably Recurrent Neural Networks (RNNs) [31], which excelled in sequential data processing, laying the groundwork for sophisticated text understanding and generation. The evolution continued into the mid-2010s with the emergence of pre-trained models, paving the way for the development of Large Language Models (LLMs). A transformative moment occurred around 2017 with the introduction of transformer models in the "Attention Is All You Need" paper [71], which for the first time relied solely on self-attention mechanisms, eliminating the need for RNNs or convolutional layers. This marked a significant paradigm shift from earlier models that combined self-attention with other architectures, to a new era where self-attention alone could drive deep, contextually aware language understanding, propelling the field of NLP into new frontiers of capability and flexibility.

Sequential models, particularly Recurrent Neural Networks (RNNs) and their advanced variants like Long Short-Term Memory (LSTM) networks, have been pivotal in processing language data due to their inherent design to handle sequential information. They process text one token at a time, maintaining a hidden state that theoretically encapsulates the information from all previously seen tokens, thus creating

a sense of memory. However, these models often struggle with long-range dependencies due to issues like vanishing or exploding gradients.

A pivotal feature of the Transformer architecture is its self-attention mechanism, which significantly enhances the model's ability to address the limitations inherent in sequential processing. Unlike sequential models that process data in order, self-attention allows the model to weigh and relate any two tokens in the input, regardless of their positions. This mechanism computes the representation of each token by considering how it relates to every other token in the sequence, enabling the model to capture dependencies without being constrained by the distance between tokens. As a result, self-attention provides a more flexible and context-aware way to encode the semantics of the input sequence, enhancing the model's ability to understand and generate language with greater nuance and coherence.

The self-attention mechanism within large language models represents a paradigm shift in how models ascertain and encode the relationships between tokens in a sequence. Specifically, self-attention calculates alignment scores, typically through a scaled dot-product attention process (Fig. 2 left), where each token's representation is dynamically influenced by computing how much focus it should place on every other token in the sequence. These scores determine the extent to which each token should attend to every other token across the sequence, enabling the weighted aggregation of token representations that effectively contextualizes each word within its surrounding linguistic environment.

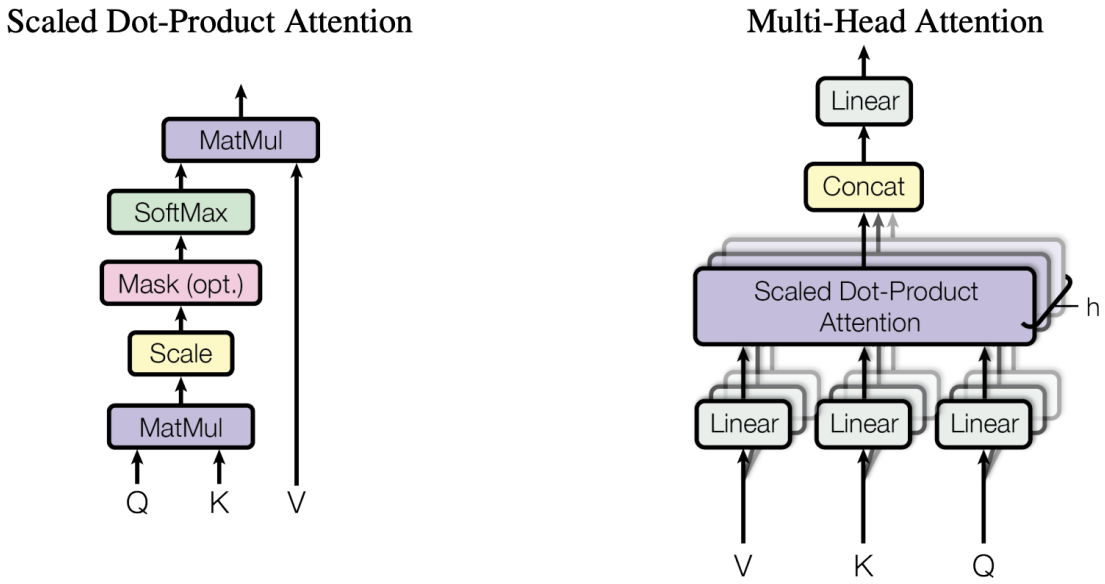



Figure 1: Illustration of the Scaled Dot-Product Attention and Multi-Head Attention mechanisms, adapted from [71]. The left part of the image demonstrates the computation within a single attention head, focusing on the scaled dot-product attention process, while the right part delineates the aggregation of multiple such heads in the multi-head attention framework.

Diving deeper into the multi-head attention feature (Fig. 1 right), we understand that it subdivides the attention mechanism into multiple "heads," allowing the model

to concurrently explore different dimensions of the data. Each head can potentially capture distinct types of relationships, such as varying syntactic or semantic connections, by computing separate sets of alignment scores. After this parallel processing, the diverse outputs from all heads are concatenated and linearly transformed, yielding a rich, integrated representation that encapsulates various linguistic nuances and facets, thereby enhancing the model's linguistic comprehension.

The attention mechanism inherent in transformer models not only enhances their performance but also offers a level of interpretability that is crucial for understanding their inner workings. Recent works [72, 83] have delved into exploring these models' attention mechanisms through visualization techniques, shedding light on how these models process and prioritize different parts of the input data.

In Figure 2, we observe a visualization that elucidates the functionality of two attention heads processing the word 'its' within an input sequence. It becomes clear from this visualization that these attention heads are adept at anaphora resolution, showcasing their ability to link referential expressions with their antecedents effectively. Additionally, Figure 3 provides insights to the scaled dot-product attention mechanism within a single attention head, focused on the token '.'. This visualization intuitively demonstrates how the attention values evolve, emphasizing a pattern where attention diminishes as the distance from the sentence-ending token increases.

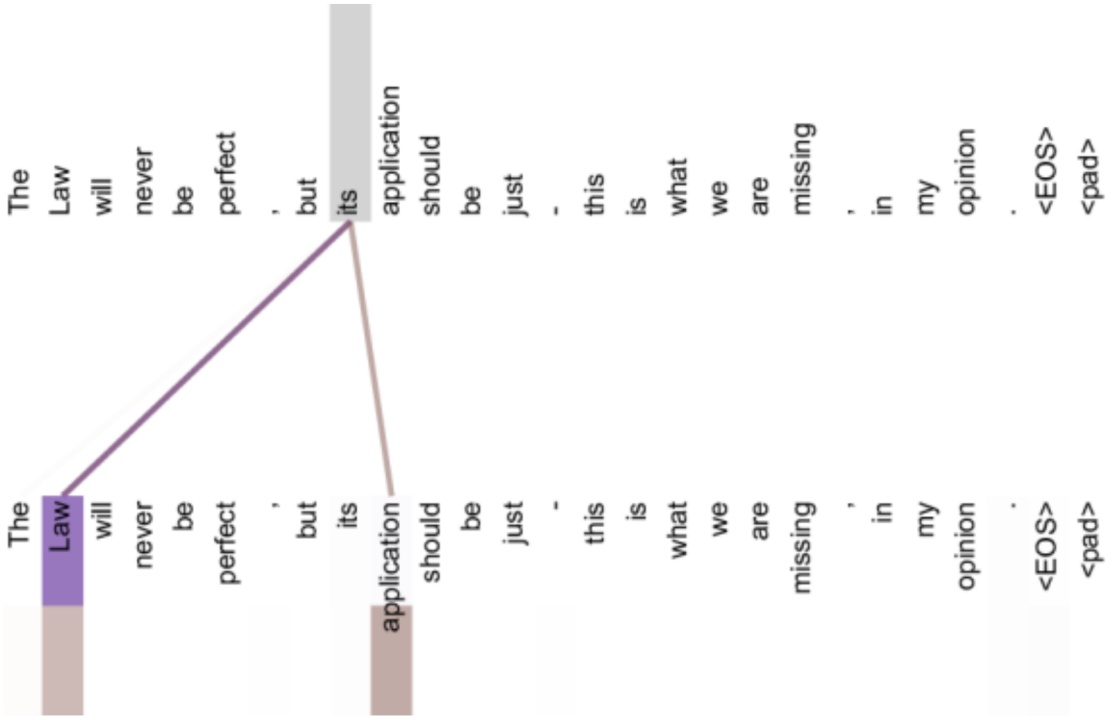



Figure 2: Illustration , adapted from [71]. Isolated attentions from just the word 'its' for attention heads 5 and 6. Note that the attentions are very sharp for this word 'Law' .

Typically, a model might employ 8 or 16 attention heads, with the choice of this parameter influencing the granularity and scope of relationships the model can capture, thereby affecting its performance and interpretability [72, 83]. This feature contrasts

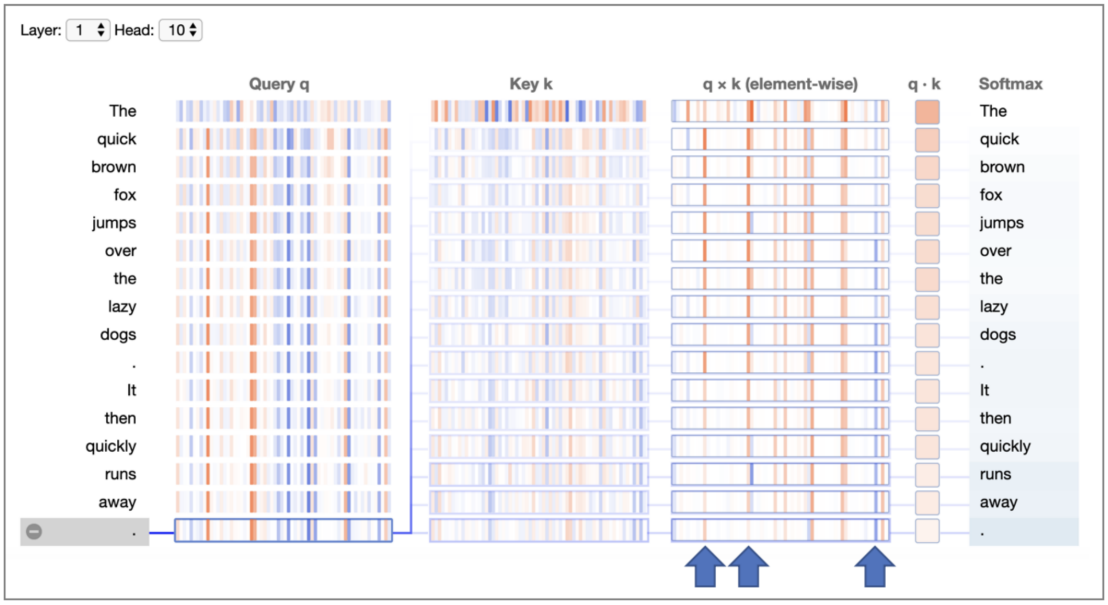



Figure 3: Illustration adapted from [72]. Neuron view of GPT-2 for layer 1, head 10 with last token selected. Positive and negative values are colored blue and orange, respectively, with color saturation based on magnitude of the value. As with the attention-head view, connecting lines are weighted based on attention between the words. Blue arrows mark positions in the element-wise products where values decrease with increasing distance from the source token (becoming darker orange or lighter blue).

with the sequential nature of RNNs, where information processing is inherently linear and time-dependent. Self-attention's parallelizable structure not only mitigates these limitations but also facilitates significantly faster training times and improved handling of longer sequences.

In the context of NLP tasks, these architectural innovations have proven instrumental across a range of applications, from translation to content generation, underpinning the versatile and powerful capabilities of large language models. However, it is also crucial to acknowledge inherent challenges, such as increased computational demands and potential difficulties in model interpretability, particularly in dissecting the specific contributions of individual attention heads or understanding the model's decision-making process in detail. Acknowledging these limitations alongside the benefits provides a comprehensive understanding of the self-attention and multi-head mechanisms within the larger narrative of language model evolution.

Following the introduction of transformer models, a notable development in the field of NLP is the application of Reinforcement Learning from Human Feedback (RLHF) to enhance the performance of large language models. The study by Ziegler et al. [93] exemplifies this approach, applying RLHF to refine pre-trained language models for specific tasks like text continuation and summarization, informed by human feedback.

In their research, Ziegler et al. [93] demonstrate how RLHF can be used to adjust language models based on human evaluations, aiming to produce outputs that are

more aligned with human judgments of quality and relevance. This process involves training a reward model from human preferences, which is then used to guide the fine-tuning of language models, enhancing their ability to generate text that resonates more aligned to with human readers preference. The integration of RLHF represents a methodological advancement in the field, enabling language models to evolve based on direct human input, thus improving their applicability and effectiveness in real-world tasks. This development underscores the shift toward more interactive and adaptive systems in NLP, where human feedback plays a vital role in refining AI outputs. Through such advancements, language models continue to evolve, offering more sophisticated tools for a variety of NLP applications, illustrating a commitment to aligning AI systems more closely with human expectations and standards, and showcasing the potential for these models to become more nuanced and context-aware in their language generation capabilities.

## 2.2 Software Engineering

Software Engineering (SE) emerged as a distinct discipline during a pivotal period in the mid-20th century, marked by the rapid expansion of computing technology. As hardware capabilities advanced, significant disparities in software development methodologies became evident, leading to what is known as the "software crisis." This crisis highlighted difficulties in developing reliable, maintainable, and efficient software within acceptable timeframes and budgets. In response, the seminal NATO conferences of the late 1960s were convened. These conferences played a crucial role in defining SE as a critical field aimed at addressing the complexities of software creation. Due to its abstract nature, software allows for substantial customization and wide applicability but also introduces significant complexities that challenge the automation of development processes. This dynamic requires a careful balance between specialized solutions and broadly applicable approaches in software design, underscoring the nuanced and continuously evolving nature of the field. The discipline of SE continues to adapt, applying established principles to new challenges across a diverse range of application domains [59, 2].

SE orchestrates all aspects of software production by applying engineering principles to develop functional, dependable, and efficient software within financial and temporal constraints. Unlike computer science, which explores theoretical aspects, or systems engineering, which addresses broader systems issues, SE focuses specifically on software solutions. It incorporates the Software Development Lifecycle (SDLC), a systematic process that includes phases such as planning, design, implementation, testing, deployment, and maintenance. This lifecycle helps manage and streamline the creation and maintenance of software, accommodating a diverse array of software types and usage scenarios. While no single methodology fits every project, leading practices vary from stand-alone applications to embedded systems. However, principles like effective process management, dependability, clear requirement gathering, and efficient resource use remain universal. These principles ensure the delivery of reliable software that meets various user and market demands, amidst ongoing technological and business challenges [67].

## 2.3 Large Language Models for Software Engineering Tasks

Building on the advancements in artificial intelligence research within SE, such as the optimization techniques employed in Search-Based Software Engineering (SBSE) which also involved natural language processing [69, 35], an intriguing new focus has emerged: the application of Large Language Models (LLMs) in SE. This area explores the potential of LLMs to manage tasks that involve understanding and generating human language, now directed toward the complexities of software development processes.

Since 2019, LLMs have garnered significant attention for their applications in SE, reflecting a broader trend across computer science disciplines. According to Fan et al. (2023), this focus has led to the recognition of LLM-based SE as an emerging subdiscipline [12]. The authors highlight the substantial growth in publications related to LLMs in SE, demonstrating the community's rapid adoption of this technology.

Supporting this observation, Hou et al. [20] document an exponential increase in publications that integrate LLMs with SE tasks—from 7 papers in 2020 to an impressive 160 in just the first half of 2023 [20]. This surge in research activity has spurred the development of the term Large Language Models for Software Engineering (LLM4SE), defining this vibrant area of study. According to Fan et al. [12], since 2022, more than 10% of all LLM-related publications have focused specifically on SE applications, underscoring the significant interest in this field [12].

Additionally, Fan et al. [12] also notes that while LLMs are extensively explored in the domain of software development, certain subdomains such as Requirements Engineering and Design, and Refactoring are notably under-represented. These areas, which rely heavily upon linguistic forms of analysis and the recognition and prediction of patterns, are identified as surprisingly ripe for consideration, presenting an opportunity for future research and application of LLMs [12].

From the detailed analysis by Hou et al. [20], it is evident that LLMs are currently being utilized across a spectrum of SE tasks. They are categorized based on their architecture into encoder-only, encoder-decoder, and decoder-only models, with the latter being the most utilized in SE, particularly for tasks involving code generation and program repair. The popularity of decoder-only models in SE underscores their effectiveness in tasks requiring deep syntactic and semantic understanding of code. This effectiveness is attributed to the autoregressive nature of decoder-only models, which allows them to generate sequences of text by predicting the next token based on the context of the preceding tokens. This capability is crucial for code generation and repair tasks, where understanding the sequential and contextual relationships within the code is essential for producing accurate and coherent outputs.

Data handling is another area of focus; the predominant use of open-source datasets, which constitute approximately 59.35% of cases, shows a reliance on widely accessible resources for training these models. Data types within these datasets are primarily code-based and text-based, aligning with the strengths of LLMs in handling complex structured data, which is crucial for SE tasks.

Moreover, Hou et al. [20] outlines specific optimization techniques like fine-tuning and the Adam optimizer, which are commonly used to enhance the performance of LLMs in SE. The use of prompt engineering, particularly in data-scarce scenarios, fur-

ther enhances these models' adaptability and effectiveness, allowing them to perform optimally across various SE tasks.

# 3 Related Work and Research Corpora

## 3.1 Research Corpora

This thesis builds upon a comprehensive literature review detailed in the foundational paper by Hou et al. [20]. This pivotal review thoroughly explored the broader application of LLMs in SE, examining the types of LLMs utilized, data handling methods, optimization and evaluation techniques employed, and the specific tasks these models have been applied to. By delving deeper into the methodology and findings of this foundational literature review, this subsection provides crucial insights into the landscape from which this research sources its corpus.

The identification process in the basis paper [20] followed a carefully structured approach, adhering to the SLR methodologies established by Kitchenham et al. [28, 27].

The five-step seach strategy started by selecting six prominent SE venues—ICSE, ESEC/FSE, ASE, ISSTA, TOSEM, and TSE—for manual searches and extended their scope to include seven databases: IEEE Xplore, ACM Digital Library, ScienceDirect, Web of Science, Springer, arXiv, and DBLP for automated searches. A quasi-gold standard was developed by screening papers from manual search based on specific inclusion and exclusion criteria, followed by the creation of a search string informed by domain expertise for the automated search. The process concluded with a snowballing search, integrating findings from both manual and automated searches to ensure a comprehensive collection of pertinent studies for subsequent analysis.

The manual and automated search resulted in 164,366 papers which was reduced to a focused set of 218 foundational studies through a series of structured steps applied in the study selection process. After an initial automated filter based on paper length trimmed the count to 63,404, a thorough title, abstract, and keyword review further reduced the pool to 4,341. At this juncture, the researchers identified the publication venues to discern the source quality, which, along with subsequent deduplication and in-depth full-text reviews considering relevance and rigor, distilled the collection to 548. Additional exclusion of papers was done by applying quality assessments and manually scoring papers based on content relevance and publication type/venue quality finally narrowed it down to 218 papers. A subsequent snowballing search, executed on these selected studies, yielded an additional 11 papers, thereby establishing a final set of 229 papers after deduplication and verification.

The distribution of the 229 relevant papers identified demonstrates that 38% were published in peer-reviewed venues like ICSE, TSE, ICSME, and SANER, while 62% appeared on arXiv, reflecting the field's swift evolution and the pre-peer review status of much Large Language Models for Software Engineering (LLM4SE) research. A temporal analysis shows a significant surge in publications from 2020 to 2023, highlighting escalating interest and research activity in LLM4SE.

### 3.2 Exploration of Replication Package from Literature Review

In this section, we delineate the review process applied to the contents of the replication package[4] provided by Hou et al. [20]. of Hou et al. [20] as of February 2024. Our objective is to scrutinize the available data within the package to identify the papers that constitute the literature review from Hou et al. [20] and to discern which information about these reviewed papers is structured and accessible. This careful examination aims to uncover potential facilitators for our analysis, ensuring we leverage any structured information to augment our understanding and streamline our research process.

The package comprises five Excel sheets: *QAC*, *RQ1*, *RQ2*, *RQ3*, and *RQ4*. A preliminary review of the column names and entries revealed that these labels stand for *Quality Assurance Criteria* and *Research Questions 1-4*, each sheet containing relevant data.

Each sheet adheres to a uniform structure, presenting identifier information for the reviewed papers, including *Title*, *URL*, *Year*, *Venue*, and *Abstract*, succeeded by sheet-specific values.

**QAC:** In their inclusion criteria, the authors of [20] evaluated the quality of each review paper, as detailed in the *Study Selection* section of [20], attributing a score to 10 Quality Assurance Criteria (QAC). The *QAC* sheet encapsulates the scores for these 10 criteria.

The Quality Assurance Criteria collectively assess the relevance, methodology, clarity, and impact of the studies within the scope of software engineering tasks, specifically examining their use of large language models (LLMs), publication venue prestige, motivation clarity, technique description, experimental detail, findings confirmation, discussion on contributions and limitations, and overall contribution to the academic or industrial community.

The sheets *RQ1* to *RQ4* provide structured documentation of the data extracted by [20] to address their respective Research Questions.

**RQ1:** This sheet includes information on the *utilized large language models (LLMs)* and their respective *Transformer Types* for each study reviewed. Columns are designated with names of *LLM Models*, covering all within the GPT and BERT series, and feature three additional columns to identify Transformer Types: Decoder only, Encoder only, or Decoder-Encoder. The inclusion of a model in a study is indicated by a column value of '1' or detailed further with specific information, such as the model's number of parameters.

**RQ2:** This sheet organizes dataset-specific details, identifying the *source* of the dataset as *open-source*, *collected*, *constructed*, or an *industry dataset*. It also categorizes the *type* of data utilized in the studies, distinguishing among *text-based*, *code-based*, *graph-based*, *software-repository-based*, or *combined* datasets. The *input types* for LLMs are outlined as *token-based*, *pixel-based*, *tree/graph-based*, or *hybrid* input types, with a '1' denoting usage or additional details provided. A column on *data preprocessing* lists specific preprocessing steps applied.

**RQ3:** This sheet provides information on the methods and metrics employed for

[4] https://docs.google.com/spreadsheets/d/1iomMvoDL2znNDQ_J4aGnqb3BhZpEMlfz/edit#gid=1471652439

optimizing and evaluating the large language models (LLMs) as described in the research studies. It includes a column for *Parameter Optimization Algorithms*, indicating the techniques utilized, such as *hyperparameter tuning*, *fine-tuning*, or a combination of both. It also details the specific optimizers used, for instance, *Adam*, *AdamW*, or *ZeRO*. For selected studies, it documents strategies to prevent overfitting within the *combat overfitting* column, mentioning approaches like *early stopping* and *data augmentation*. The sheet categorizes the machine learning problem in the *problem type* column into *classification*, *regression*, *generation*, or *recommendation*. Additionally, a *Metrics* column records the metrics applied by the research papers to assess their findings.

**RQ4:** This sheet organizes the research papers according to SE-Activities and SE-Tasks. It features a dedicated column for each SE-Activity: *Software Requirements*, *Software Design*, *Software Development*, *Software Testing*, *Software Management*, and *Software Maintenance*. Within these columns, the corresponding *SE-Task(s)* addressed by the research are listed as values. It is observed that some papers could not be associated with a specific SE-Task, marked by the value *other*. Additionally, there are instances where certain studies are not aligned with any SE-Activity.

Upon reviewing the list of paper titles retrieved, the paper [91] titled *Large language models are human-level prompt engineers* stood out as it seemed unrelated to the application of LLMs in addressing a Software Engineering task. This paper is listed in Table 6 of [20] among papers using *Text-based datasets* and in Table 9 among those using *Text as Tokens* as input for the LLMs.

However, upon closer examination of the content, it was found to violate the inclusion criterion 2) in Table 3, which states, *"The paper claims that the study involves an SE task,"* of the survey paper [20]. It appears this paper uses the SE task of *Program Synthesis* to address the topic of automatic prompt engineering for LLMs. For this reason, this paper is excluded from the investigation in this work.

# 4 Method

This section presents the methodological framework applied to prepare and conduct the review, which yielded qualitative summaries of the selected papers, quantitative scores for the reliability and relevance of their evaluations, and insights into the limitations and strengths of their methods. These results form the basis for the critical analysis provided in Section 5. In the first subsection (Section 4.1), the overall six-phase framework used to address the study's research questions is described. The second subsection (Section 4.2) details the preparation steps taken to set the stage for the review, and the third subsection (Section 4.3.1) explains how the review was conducted, including the procedures for analyzing each paper's evaluation strategy.

## 4.1 Summary of the Study's Approach

This subsection provides a high-level summary of the systematic approach employed in this study to critically review and evaluate research papers leveraging Large Language Models (LLMs) for Software Engineering (SE) tasks. The approach is structured into six phases, designed to ensure a comprehensive and focused analysis.

Phases 1 to 3 (4.1.2-4.1.3) lay the groundwork by identifying relevant papers, defining key evaluation aspects, refining the scope to ensure manageability, and grouping the papers by their task objectives for thematic coherence. Phase 4 (4.1.5) details the review process, summarizing evaluation strategies, scoring reliability and relevance, and identifying strengths and limitations based on the defined evaluation aspects. In Phase 5, the findings are described groupwise, with a critical analysis of strengths and limitations across multiple dimensions. Finally, Phase 6 synthesizes the findings, discussing reliability and relevance from different perspectives and linking them back to the research questions.

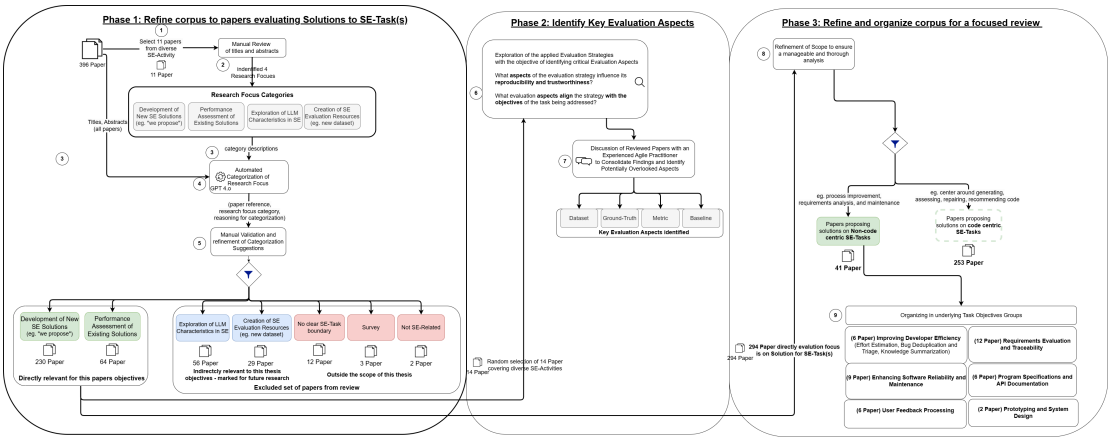



Figure 4: Flow diagram illustrating the steps from Phase 1 to Phase 3, outlining the process for preparing the review

### 4.1.1 Core Evaluation Dimensions: Reliability and Relevance

The assessment of evaluation strategies in this study is grounded in the dimensions of reliability and relevance, which are defined and discussed in detail in Section 4.2.2. Both dimensions are scored on a scale from 0 to 5, providing a structured framework for evaluating the reproducibility and trustworthiness of the methodologies (reliability) and their alignment with the real-world needs of the SE task(s) addressed (relevance) in LLM-based SE research.

These dimensions directly align with the study's research questions (1.2), addressing **RQ1** by evaluating the reliability of current practices and **RQ2** by examining their relevance to real-world software engineering needs. Together, they provide a foundation for identifying and analyzing the key gaps and limitations explored in *RQ3*.

### 4.1.2 Phase 1: Categorization by Research Focus

The goal of this phase is to categorize the research papers into distinct groups based on their research focus and identify those most relevant to this thesis. Specifically, the focus is on papers that either propose *novel solutions to SE tasks* or evaluate *existing solutions for SE tasks*, as these align with the thesis's objective of critically assessing the evaluation strategies applied to the SE tasks being addressed. At the end of this process, 295 papers out of an initial set of 396 were identified as directly relevant

to this thesis. Papers that explore LLM characteristics or create resources such as datasets or benchmarks are categorized but not analyzed further, as they fall outside the primary scope of this thesis.

This phase is carried out in the following steps (see Section 4.2.1 for more details):

1. **Initial Manual Selection:** A subset of papers is selected, ensuring diversity in SE activities such as code generation, bug detection, and test generation. This subset is then manually reviewed to guide the identification of research focus categories.

2. **Definition of Research Focus Categories:** Four categories are identified:

   (1) Development of Novel SE Solutions.

   (2) Performance Assessment of Existing Solutions.

   (3) Exploration of LLM Characteristics.

   (4) Creation of SE Evaluation Resources.

   The first two categories—*Development of Novel SE Solutions* and *Performance Assessment of Existing Solutions*—are directly relevant to the thesis and prioritized for further analysis. Papers in the remaining two categories are categorized but not analyzed in detail, as they fall outside the thesis's primary focus.

3. **Automated Categorization:** An LLM is used to categorize all papers into these groups based on their titles and abstracts.

4. **Manual Review and Corrections:** The automated results are reviewed and corrected to ensure accuracy and consistency. During this process, papers falling outside the scope of this thesis are identified and excluded. These include:

   - Papers proposing solutions with no clear SE-task boundaries.
   - Survey papers.
   - Papers not related to SE tasks.

   While these exclusions are separate from the four research focus categories, they help refine the dataset to align with the thesis's objective.

By systematically categorizing the papers, this phase ensures a focused dataset for subsequent phases of the study, prioritizing papers that align with the thesis's goal of evaluating LLM-based solutions in SE contexts.

### 4.1.3 Phase 2: Identification of Key Aspects of Evaluation through Detailed Review

The primary goal of this phase is to identify the key aspects of evaluation strategies used in LLM-based SE research. To achieve this, a representative subset of 14 papers was selected from the 295 relevant papers categorized in Phase 1. These papers were chosen to capture a diverse range of SE tasks and machine learning applications, such as requirements engineering, code generation, and bug detection, ensuring broad

coverage of evaluation practices. This subset served as a basis for understanding evaluation strategies across various SE contexts.

During the detailed review of these 14 papers, it became evident that analyzing all 295 papers would be beyond the practical scope of this thesis. This realization informed the decision to narrow the thesis's focus in subsequent phases to ensure depth and rigor.

The detailed review of the subset focused on identifying key evaluation aspects, including:

- **Datasets:** The datasets used and their alignment with real-world SE challenges.
- **Ground Truths:** The clarity and appropriateness of the ground truths used.
- **Benchmarks:** The role of benchmarks in comparative evaluation.
- **Metrics:** The suitability of metrics for measuring performance.

These aspects are critical for assessing the *reliability* and *relevance* of evaluation methods and provide the foundation for the scoring process in later phases. For a detailed description of this phase, see Section 4.2.3.

### 4.1.4 Phase 3: Refinement of Scope and Grouping by Task Objectives

The goal of this phase is to refine the scope of the thesis and group papers by their underlying objectives to enable a systematic and focused evaluation. This approach ensures depth and rigor in analyzing evaluation strategies, aligning with the thesis's practical constraints and its emphasis on non-code-centric SE tasks.

To achieve this, the focus was narrowed to SE tasks such as requirements engineering, software specifications, API documentation, and bug detection and reproducibility. Code-centric evaluations (e.g., code generation and code completion), while prevalent in the corpus, presented unique challenges that require dedicated investigation in future work. This refinement aligns with the research emphasis of DLR, the primary stakeholder, and ensures the thesis's scope remains manageable.

Papers were grouped based on their underlying objectives, with the rationale that papers sharing similar objectives address common strengths and challenges relevant to their respective tasks. This thematic grouping enabled consistency and focus within each group during the review process. While some papers align with multiple groups to varying degrees, the grouping allowed for a structured comparative analysis.

By refining the scope and grouping papers based on their objectives, this phase structured the dataset for subsequent comparative analyses. These analyses address the research questions outlined in Section 1.2, ensuring that the results and discussion sections provide actionable insights into evaluation practices.

### 4.1.5 Phase 4: Group-wise review - Evaluation summarization and scoring

The goal of this phase was to analyze the evaluation strategies employed in the selected 41 papers systematically. This analysis aimed to produce qualitative summaries of each paper's evaluation reliability and relevance and to assign quantitative scores for these dimensions. These outputs provide the groundwork for subsequent critical

analysis of limitations across the key evaluation aspects (Section 5.2) and statistical analyses of reliablity and relevance of the evaluation practices (Section 5.3).

The review was conducted group-wise, aligning with the task objectives identified in Phase 3, to ensure coherence and consistency in the analysis of papers with similar goals. Each paper was reviewed along the key evaluation aspects identified earlier, including datasets, baselines, metrics, ground truth, and validation methods. Observations of advanced practices, such as cross-validation techniques and error analyses, led to the inclusion of a new evaluation aspect, *Validation Method*, for systematic documentation.

For each paper, summaries of strengths and limitations in terms of reliability and relevance were documented, alongside scores ranging from 0 to 5 for these dimensions. The review process included an in-depth analysis of sections such as the title, abstract, and introduction to understand the paper's contributions. Particular focus was given to methods, tables, and figures detailing datasets, metrics, baselines, and validation practices. For instance, papers frequently contained specific sections on datasets—often including ground truth creation details—and metrics or baselines, which facilitated structured and consistent analysis.

This phase is detailed in Section 4.3.1, while descriptive summaries of grouped evaluations are presented in Section 5.1. These structured outputs laid the foundation for the critical and statistical analyses conducted in subsequent phases.

#### 4.1.6 Phase 5: Critical Analysis of Strengths and Limitations of Evaluation Strategies

The goal of this phase was to conduct a critical analysis of the evaluation strategies identified during the group-wise review (Phase 4), focusing on their strengths and limitations across key evaluation aspects. The phase aimed to identify recurring patterns, gaps and insights for addressing the research questions. The outcomes of this phase include a thematic understanding of challenges and notable practices in datasets, baselines, metrics, ground truth, and validation methods, as well as insights into how these aspects vary across task objectives, research focuses, ML task types, and temporal trends.

The findings and insights generated during this phase serve as foundational inputs for subsequent analyses and interpretations.

- **Categorized Findings:** Evaluation strategies were critically analyzed across:
  - *Task objective groups* (e.g., improving developer efficiency, enhancing software reliability).
  - *Research focuses* (Development of New SE Solutions, Performance Assessment of Existing Solutions).
  - *ML task types* (classification, regression, recommendation, generation).
  - *Temporal trends* (evaluation practices from 2020–2024).
- **Identification of Patterns and Gaps:** Recurring limitations such as dataset representativeness and generalizability, inadequate baselines, and inconsistent validation methods were highlighted.

For details on the critical analysis process and findings, refer to Sections 5.2 through 5.2.4.

### Phase 6: Analyzing Reliability and Relevance from Multiple Perspectives

The goal of this phase was to synthesize the findings from previous analyses, providing a holistic assessment of the reliability and relevance of evaluation strategies across multiple perspectives. The results include descriptive summaries, identified trends, and actionable insights into the recurring challenges and strengths in evaluation methodologies. This phase also sets the stage for answering the study's research questions.

- **Overall Reliability and Relevance:** The analysis reveals moderate performance across reliability and relevance dimensions, with recurring limitations in dataset generalizablity and representativeness, baseline diversity, metrics suitability, validation scope, and ground truth clarity. These insights are quantitatively summarized in Sections 5.3.1 and 5.3.2.
- **Cluster Patterns:** Reliability and relevance scores were analyzed to identify distinct clusters and outliers, highlighting systemic trends and task-specific differences in evaluation strategies (see Section 5.3.2).
- **Task-Specific Contrasts:** Comparative analyses of differing evaluation practices within the same task were conducted, focusing on papers with varying reliability and relevance scores within the same task. This analysis provided nuanced insights into how methodological choices impact the quality of evaluation strategies and emphasized the need for task-tailored approaches to address recurring weaknesses. These contrasts were explored across multiple tasks.

The combined findings from Phases 4, 5, and 6 form the foundational inputs for answering the research questions. Details of this synthesis and its connection to the research questions are presented in Sections 6.1, 6.2, and 6.3. An overview of the research questions is available in Section 1.2.

## 4.2 Setting the Stage for the Review

This subsection establishes the methodological foundation for the critical review of evaluation strategies in research leveraging Large Language Models (LLMs) for Software Engineering (SE) tasks. The goal is to outline the systematic process adopted to identify, categorize, and refine the scope of studies, ensuring a focused and manageable review aligned with the thesis objectives.

The scope was refined in two stages: first, by excluding papers whose research focus (4.2.1) is not directly on evaluating a proposed or existing solution, but which may still serve as valuable candidates for future work, such as studies exploring LLM characteristics or creating datasets and benchmarks; and second, by narrowing the focus to non-code-centric Software Engineering tasks (4.2.4), such as requirements engineering, software specifications, API documentation, and bug detection. Code-centric tasks, including code generation and completion, were excluded from detailed

analysis due to their distinct challenges and evaluation requirements, which warrant dedicated future work. These steps ensured that the review remained both rigorous and manageable.

To support a structured and in-depth analysis, this section identifies key evaluation aspects (4.2.3)—such as datasets, ground truths, metrics, and baselines—that underpin the evaluation of reliability and relevance (4.2.2). Additionally, the selected papers were grouped based on their underlying task objectives (4.2.5), ensuring thematic coherence and enabling meaningful comparisons across task-specific evaluation strategies.

### 4.2.1 Research Focus Categorization

The primary objective of this phase is to identify research papers where the evaluation focuses on assessing LLM-based solutions to SE problems. Of the initial corpus of 396 papers, **17 papers were excluded**, resulting in a refined set of **379 papers**. These exclusions were made due to misalignment with the scope of this research, such as being surveys, lacking a focus on specific SE tasks, or addressing areas outside of software engineering. Additionally, six papers were categorized under multiple research focus categories, reflecting their dual contributions to both proposing new solutions and creating relevant datasets or evaluation methods.

From the remaining 379 papers, **294 fall into the research focus categories of *Development of New SE Solutions* and *Performance Assessment of Existing Solutions***, which are directly relevant to the goals of this thesis. The remaining papers were categorized as either *Exploration of LLM Characteristics in SE* or *Creation of SE Evaluation Resources* but were not analyzed further due to their indirect relevance to the thesis.

#### Process of Research Focus Category Identification

This categorization process was executed in the following steps, blending a manual review of papers with an automated categorization using an LLM-based model. The process was structured as follows.

**1. Initial Manual Selection of Papers:** A random subset of 11 papers was initially selected, ensuring that the selected papers represented a diverse set of SE activities [19, 1, 8, 75, 79, 54, 6, 34, 37, 61, 21]. These papers covered a wide range of SE-Activities including code generation, bug detection and vulnerability detection.

**2. Review of Titles and Abstracts:** The titles and abstracts of these 11 papers were manually reviewed to answer the following questions:

- **What is the contribution of the paper?** This could include fine-tuning a model, creating a new framework, empirical observations, or creating datasets.
- **What is the object of evaluation?** Whether the evaluation focuses on the proposed solution, empirical observations, or the assessment of a created resource.

- **Who is the target group benefiting from the paper's contributions?** Potential target groups include end users, such as developers benefiting from SE solutions, and researchers who gain insights from empirical investigations or make use of resources like new datasets or benchmarks.

This manual review helped in identifying distinct research focuses, which were then used to categorize the remaining papers.

**3. Identification of Categories:** Based on the manual review, four distinct categories of research focus were identified:

- **Development of New SE Solutions**: Papers proposing novel (own) LLM-based solutions for SE tasks (e.g., code generation or bug detection).
- **Performance Assessment of Existing Solutions**: Papers empirically evaluating the performance of existing LLM-based SE solutions.
- **Exploration of LLM Characteristics in SE**: Papers that investigate properties of LLMs, such as robustness or determinism, in relation to SE tasks.
- **Creation of SE Evaluation Resources**: Papers focused on developing resources like datasets or benchmarks for future evaluations.

These categories reflect the diversity of research in LLM usage for SE and provide a structured framework for reviewing and evaluating papers.

**4. Automated Categorization using an LLM:** After establishing the categories, an LLM (GPT-4o model) was used to automatically categorize the remaining papers in the dataset. The titles and abstracts were extracted from the replication package[5] provided by Hou et al. [20].

The model was instructed to classify each paper into one of the identified categories or the *"other"* category if none of the pre-defined categories applied. The LLM was also asked to provide reasoning for its classification.

This categorization process was implemented using the OpenAI API[6] and the LangChain library to extract structured output. The specific prompts and API interactions are provided in the appendix A.

**5. Manual Review and Corrections:** After the automated categorization, the author of this thesis manually reviewed the LLM-generated categorizations for accuracy and consistency. The review process focused on papers that appeared misclassified or unclear based on their title and abstract. The manual review resulted in two distinct outcomes: refinements to the categorization of papers and the exclusion of papers that fell outside the scope of this thesis.

[5] https://docs.google.com/spreadsheets/d/1iomMvoDL2znNDQ_J4aGnqb3BhZpEMlfz/edit#gid=1471652439

[6] https://openai.com/api/

**Category Refinements:** Five papers were categorized under both *Development of New SE Solutions* and *Creation of SE Evaluation Resources*. Four of these papers contributed new datasets in addition to their primary focus on developing SE solutions [65, 25, 70, 13]. The remaining paper [36] was categorized under both categories as it introduced a novel evaluation method alongside the proposed solution.

**Paper Exclusions:** The following papers were excluded from the review for reasons detailed below:

- **Surveys and reviews**: Papers initially categorized under *Performance Assessment of Existing Solutions* were excluded upon further review as they were surveys or reviews without a clear focus on proposing a solution to SE tasks [46, 74].
- **Lack of specific SE task focus or clear task boundaries**: These papers categorized under *Development of New SE Solutions* or *Performance Assessment of Existing Solutions* were excluded upon closer manual review because they did not focus on a specific SE task or lacked clear task boundaries [62, 53, 57, 88, 63, 55, 4, 33, 58, 64, 90, 68].
- **Unrelated to software engineering**: Three papers were excluded because they did not address software engineering:
  - [91]: Excluded because, although it utilizes the SE task of program synthesis, its focus is on solving prompt engineering for LLMs rather than addressing a specific SE task.
  - [29]: Excluded as it proposes a multilingual solution for detecting phishing sites, falling within the scope of Social Engineering rather than Software Engineering.
  - [26]: Excluded because it focuses on supporting developers with hyperparameter tuning of machine learning models, which is not an SE-specific task.

### Identified Research Focus Categories

The following subsections provide a detailed description of the four identified research focus categories.

The categorization of research papers into these four groups allows for a clear distinction between studies that directly assess LLM-based solutions for SE tasks and those that contribute indirectly by exploring LLM properties or developing evaluation resources. For the purposes of this thesis, only the first two categories—*Development of New SE Solutions* and *Performance Assessment of Existing Solutions*—are considered directly relevant and are the focus of the subsequent review of evaluation strategies. While categories *Exploration of LLM Characteristics in SE* and *Creation of SE Evaluation Resources* provide useful context for future work, they fall outside the specific goal of this research, which is to critically analyze the evaluation methods used to assess LLM-based SE solutions.

### Development of New SE Solutions

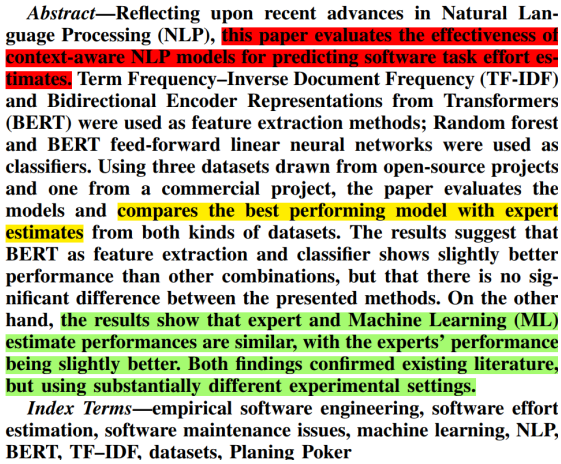
This category includes 230 studies which focus on developing new solutions to SE problems using LLMs. These papers typically introduce novel frameworks, fine-tuned models, or methods to address specific SE tasks such as code generation, bug detection, program repair, or test generation.

- **Evaluation focus**: The evaluation methods in these papers are aimed at directly assessing the effectiveness of the proposed solutions in solving the specific SE problem(s).
- **Relevance**: This category is directly relevant to this thesis, as it aligns with the goal of assessing the reliability and relevance of evaluation strategies for LLM-based SE solutions.

### Performance Assessment of Existing Solutions

This category encompasses 64 papers that focus on evaluating existing LLM-based solutions for SE tasks. Rather than proposing new methods, these studies assess and validate the performance of established LLM solutions in real-world SE applications.

- **Evaluation focus**: These papers aim to measure the effectiveness of existing solutions through comparative analyses and benchmarking.
- **Relevance**: This category is also directly relevant to the thesis, as it provides insight into how well current evaluation strategies assess the performance of existing LLM-based solutions in SE tasks.



***Abstract*—Reflecting upon recent advances in Natural Language Processing (NLP), this paper evaluates the effectiveness of context-aware NLP models for predicting software task effort estimates. Term Frequency–Inverse Document Frequency (TF-IDF) and Bidirectional Encoder Representations from Transformers (BERT) were used as feature extraction methods; Random forest and BERT feed-forward linear neural networks were used as classifiers. Using three datasets drawn from open-source projects and one from a commercial project, the paper evaluates the models and compares the best performing model with expert estimates from both kinds of datasets. The results suggest that BERT as feature extraction and classifier shows slightly better performance than other combinations, but that there is no significant difference between the presented methods. On the other hand, the results show that expert and Machine Learning (ML) estimate performances are similar, with the experts' performance being slightly better. Both findings confirmed existing literature, but using substantially different experimental settings.**

***Index Terms*—empirical software engineering, software effort estimation, software maintenance issues, machine learning, NLP, BERT, TF–IDF, datasets, Planing Poker**

Figure 5: Example abstract of a study assessing performance of existing solutions

The marked sections in the abstract of [1] highlight the study's focus:

The red-highlighted text indicates the study's primary objective of **evaluating the effectiveness** of context-aware NLP models, specifically BERT and TF-IDF, for predicting software task effort estimates. This reflects a focus on assessing **existing solutions** rather than introducing new methods or frameworks. The yellow-highlighted text

emphasizes the study's **comparative analysis** between machine learning (ML) models and expert estimates across multiple datasets, including both open-source and commercial projects. This aspect showcases the study's intent to benchmark existing solutions against human performance, further underlining its evaluative nature. The green-highlighted text summarizes the **empirical findings**, such as the comparable performance of ML estimates and expert estimates. The results validate and extend existing literature under different experimental conditions, reinforcing the study's focus on **assessing the reliability of current approaches** within diverse software engineering contexts.

#### Exploration of LLM Characteristics in SE

This category, comprising 56 papers, investigates specific properties of LLMs in relation to SE tasks. These studies explore attributes such as robustness, generalizability, and interpretability, with the goal of understanding how these characteristics impact SE applications.

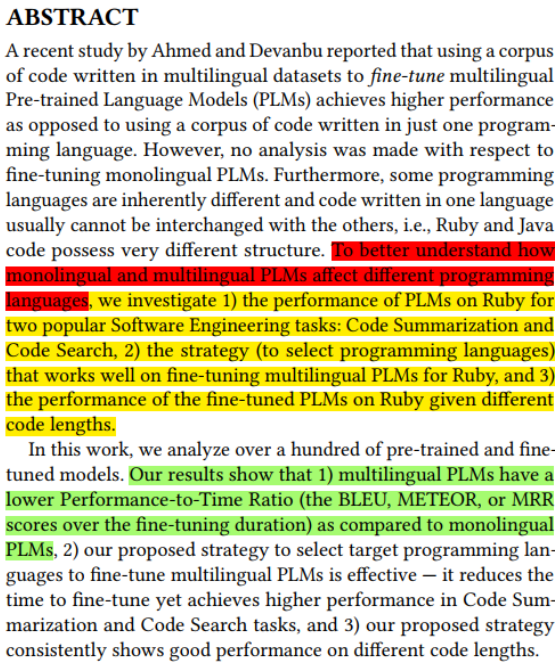
- **Evaluation focus**: The evaluation methods in this category are primarily designed to support research questions or assumptions regarding LLM properties in SE contexts.
- **Relevance**: Although these papers do not directly focus on evaluating solutions to SE problems, they contribute valuable insights into how LLM properties influence SE applications. Understanding these characteristics is crucial for designing future SE solutions and developing more targeted evaluation strategies. While papers from this category are indirectly relevant to this thesis, they are outside the scope of the current work and are considered as potential objects for future research.

Figure 6 illustrates a representative example [6] of research within this category. The marked sections in the abstract highlight the study's focus:

The red-highlighted text indicates the objective of **understanding how** monolingual and multilingual PLMs differ in their handling of programming languages, which aligns with investigations of LLM robustness and adaptability in diverse contexts. The yellow-highlighted text outlines specific research aspects, such as the impact of programming language choice, fine-tuning strategies, and code lengths on PLM performance. These aspects demonstrate a focus on exploring LLM properties rather than directly solving an SE task. The green-highlighted text presents findings that provide insight into the behavior of multilingual PLMs, such as their lower Performance-to-Time Ratio compared to monolingual PLMs. Rather than solely assessing performance, this finding sheds light on how LLMs behave under different configurations and conditions.

#### Creation of SE Evaluation Resources

This category consists of 29 papers that focus on developing datasets, benchmarks, or other resources that can be used for evaluating LLM-based solutions in SE tasks.



**ABSTRACT**

A recent study by Ahmed and Devanbu reported that using a corpus of code written in multilingual datasets to *fine-tune* multilingual Pre-trained Language Models (PLMs) achieves higher performance as opposed to using a corpus of code written in just one programming language. However, no analysis was made with respect to fine-tuning monolingual PLMs. Furthermore, some programming languages are inherently different and code written in one language usually cannot be interchanged with the others, i.e., Ruby and Java code possess very different structure. To better understand how monolingual and multilingual PLMs affect different programming languages, we investigate 1) the performance of PLMs on Ruby for two popular Software Engineering tasks: Code Summarization and Code Search, 2) the strategy (to select programming languages) that works well on fine-tuning multilingual PLMs for Ruby, and 3) the performance of the fine-tuned PLMs on Ruby given different code lengths.

In this work, we analyze over a hundred of pre-trained and fine-tuned models. Our results show that 1) multilingual PLMs have a lower Performance-to-Time Ratio (the BLEU, METEOR, or MRR scores over the fine-tuning duration) as compared to monolingual PLMs, 2) our proposed strategy to select target programming languages to fine-tune multilingual PLMs is effective — it reduces the time to fine-tune yet achieves higher performance in Code Summarization and Code Search tasks, and 3) our proposed strategy consistently shows good performance on different code lengths.

Figure 6: Example abstract of a study investigating LLM properties in SE tasks

These studies contribute essential tools for future research but do not directly evaluate SE solutions themselves.

- **Evaluation focus**: These papers evaluate the quality and utility of the resources (datasets and benchmarks) they create, ensuring they are suitable for future SE evaluations.

- **Relevance**: Although indirectly relevant to this research, this category is not within the scope of the thesis.

### 4.2.2 Defining Reliability and Relevance

The two key dimensions—**reliability** and **relevance**—serve as the foundation for evaluating the *evaluation methodologies* applied in LLM-based software engineering (SE) research. These dimensions provide a structured framework for assessing how well evaluation methods align with *scientific rigor* and *real-world applicability*.

A **scoring system ranging from 0 to 5** is applied to each dimension, serving as an *indicator for identifying trends and patterns* across the reviewed studies. The definitions and criteria for these scores are outlined below, emphasizing their *interpretive nature*.

#### Reliability

**Reliability** measures the *consistency, reproducibility, and scientific rigor* of an evaluation method. A highly reliable evaluation produces *stable results across experiments, datasets, and environments*, with transparent reporting of procedures.

- **0: No Reliability** – The evaluation lacks scientific rigor and transparency; reproducibility is impossible.
- **1: Very Low Reliability** – Significant weaknesses, such as insufficient or biased datasets, undermine reproducibility.
- **2: Low Reliability** – The evaluation uses flawed or outdated datasets and metrics; reproducibility is uncertain.
- **3: Moderate Reliability** – The evaluation employs established datasets and metrics but lacks robustness in addressing task-specific nuances.
- **4: High Reliability** – The evaluation uses diverse datasets, well-defined metrics, and robust validation strategies, though minor limitations persist.
- **5: Very High Reliability** – The evaluation is exemplary, combining multiple datasets, diverse metrics, and rigorous analysis to ensure high reproducibility and consistency across scenarios.

**Relevance**

**Relevance** assesses how well an evaluation aligns with the *practical and real-world challenges* of the SE task being addressed. A highly relevant evaluation reflects *real-world data variability, edge cases, and practical constraints*.

- **0: No Relevance** – The evaluation has no meaningful alignment with the SE task's objectives.
- **1: Very Low Relevance** – The evaluation lacks connection to real-world challenges, relying on synthetic or overly simplistic data.
- **2: Low Relevance** – Partial alignment with task-specific challenges, with key real-world aspects overlooked.
- **3: Moderate Relevance** – The evaluation addresses key task-specific aspects but misses real-world edge cases or scalability concerns.
- **4: High Relevance** – Strong alignment with real-world conditions, covering diverse datasets and metrics with actionable insights.
- **5: Very High Relevance** – Comprehensive alignment with real-world challenges, including edge cases, diverse datasets, and actionable insights for practitioners.

**Scoring Interpretation and Limitations**

The **scoring system functions as an indicator** to identify *trends and patterns* across evaluation strategies rather than as an *absolute benchmark* for assessing individual studies. While the scoring criteria are clearly defined, a *degree of subjectivity remains inevitable*, particularly in borderline cases where scores might fluctuate between adjacent levels (e.g., between 3 and 4).

This subjectivity can arise due to:

- *Ambiguity in Reporting:* Insufficient details in methodology sections.
- *Reviewer Judgment:* Differences in interpretation when assigning scores in nuanced cases.
- *Granularity Limitations:* The scoring scale simplifies complex methodological nuances into discrete levels.

As a result, **individual scores should not be viewed in isolation but rather in relation to broader trends**. The scoring aims to *highlight recurring strengths and weaknesses, identify evaluation gaps, and guide improvements* in future research practices.

#### 4.2.3 Review of the First Batch and Identifying Key Aspects of Evaluation

In **Phase 1** (Section 4.2.1), we categorized research papers based on their research focus, with the aim of identifying those where the object of evaluation is a solution utilizing LLMs for solving SE-Task(s). As a result, we identified two categories directly relevant to this thesis: *Development of New Solutions* and *Performance Assessment of Existing Solutions*. From these categories, 230 papers focused on new solution development, and 65 papers focused on evaluating existing solutions, forming the basis for further investigation in this section.

In **Section 4.2.2**, we defined the key concepts of *reliability* and *relevance* in the context of evaluating LLM-based solutions for Software Engineering (SE) tasks. These definitions provide the foundation for assessing the quality of the evaluation methods applied in the selected studies.

This section now reviews a subset of 14 papers from the two most relevant categories of research focus. The primary objective is to identify key aspects of evaluation that will guide the subsequent iterative review process. These key aspects will serve as the criteria for assessing the reliability and relevance of the applied evaluation strategies across different SE tasks.

##### Distribution of SE Activities

The papers were chosen to ensure comprehensive coverage of various SE tasks, facilitating an exploration of evaluation methods for different problem domains. The selection mirrors the distribution of SE activities and tasks from Hou et al. [20]:

- **Software Development (New Solutions)**: Code generation, Code Translation and Story point estimation [75], [65], [24], [14].
- **Software Development (Existing Solutions)**: Code generation and Verilog code generation [39], [70].
- **Software Maintenance (New Solutions)**: Bug triage, duplicate bug report detection, and program repair [34], [23], [49], [25].
- **Software Maintenance (Existing Solutions)**: Program repair [13].
- **Software Quality Assessment (New Solutions)**: Unit test generation [79].

- **Software Quality Assessment (Existing Solutions)**: Unit test generation [66].
- **Software Management (Existing Solutions)**: Effort estimation for software maintenance [1].

This distribution allows for an examination of evaluation strategies across different SE contexts, providing insights into how evaluation methods adapt to varying SE tasks and real-world scenarios.

### Identifying Key Aspects of Evaluation

Throughout the review process, several important observations were made that led to the identification of key aspects critical to evaluating the reliability and relevance of the applied evaluation methods in the selected papers. These observations, combined with collaborative discussions with the reviewer of this thesis—an experienced software engineering practitioner—helped shape the identification of these aspects. The following sections outline some of the key papers that influenced the identification of these aspects and the insights gained from them.

### Collaboration and Comparative Review

Two papers—Fu and Tantithamthavorn [14] and Alhamed and Storer [1]—were reviewed and discussed in detail with the reviewer of this thesis, a member of the Intelligent and Distributed Systems department at the German Aerospace Center (DLR) and an experienced agile practitioner. Both papers addressing effort estimation tasks in software development provided contrasting approaches to key aspects such as ground truth, datasets, data preprocessing, metrics, and benchmarks.

### Fu and Tantithamthavorn [14]: GPT2SP: A Transformer-Based Agile Story Point Estimation

- **Dataset**: The study utilized data from 16 projects in well-known open-source repositories, such as Apache, Atlassian, and Moodle, with story points estimated by users as the ground truth. However, the range of story points across projects varied significantly (from 1-8 to 1-100), raising concerns about the consistency of the evaluation. Additionally, limiting the dataset to open-source projects reduces the relevance of the evaluation, as effort estimation practices can differ significantly between open-source and industry projects.
- **Ground Truth**: Fu and Tantithamthavorn [14] used story points without categorization as the ground truth. Precisely estimating time effort is a challenging task even for humans, and using non-categorized story points in a regression task reduces both the reliability and relevance of the evaluation. Standardizing and categorizing the story points into broader effort categories would have allowed for more consistent and practical assessments, as it accounts for inherent uncertainties in real-world estimations. This approach would have better aligned the evaluation with the complexities of real-world SE tasks.

- **Metrics**: The paper used Mean Absolute Error (MAE) as the evaluation metric, reporting an average MAE of 1.6. Given the variance in story point ranges, this metric required more careful interpretation to account for differences in scale, suggesting that standardization could improve reliability.
- **User Study**: The study included a small-scale user study with 16 agile practitioners to assess the real-world applicability of the solution. Although the sample size of practitioners was limited, the study contributed to the relevance of the evaluation by involving actual users. Additionally, the solution was assessed both with and without interpretability features, which further strengthened the evaluation by providing insights into the practical utility of the solution in different contexts.

**Alhamed and Storer [1]: Evaluation of Context-Aware Language Models for Effort Estimation**

- **Dataset**: This paper used a mix of four open-source datasets and one industrial dataset, enhancing the relevance of the evaluation by introducing more diverse and realistic data.
- **Ground Truth**: Unlike Fu and Tantithamthavorn [14], Alhamed and Storer [1] standardized the story points and time estimates across different projects, making the evaluation more reliable. They also categorized the story points and time estimates, inspired by the Planning Poker estimation approach. This transformation improved the relevance of the evaluation by better reflecting practical estimation tasks in real-world SE environments. Furthermore, for comparing the estimates of learned embedding-based estimators with human experts, they used realized times from project log entries. This approach further increased the relevance of the evaluation method in terms of ground truth selection, especially since the study found that human experts performed only slightly better than learned embeddings-based estimators.
- **Baseline**: The study compared Decision Trees using TF-IDF features with Decision Trees using learned embeddings, as well as with human expert estimates. The results showed that the models using learned embeddings were only slightly better than those using TF-IDF features but closely matched the performance of human experts. This comparison increased the relevance of the evaluation by demonstrating the competitiveness of learned embeddings in real-world scenarios, offering valuable insights for practitioners assessing the model's effectiveness.

These contrasting approaches in terms of dataset quality, ground truth construction, and metric selection provided insights into how different evaluation strategies can impact both the reliability and relevance of a study's findings.

**Additional Observations and Insights**

Several other papers reviewed provided further insights into key aspects of evaluation. These observations helped solidify the focus on datasets, ground truth, bench-

marks, and evaluation metrics as the central elements for assessing reliability and relevance.

- **Training Data Usage**: Both Jin et al. [25] and Isotani et al. [23] used their training data for evaluation without clearly mentioning a train-validation-test split, raising concerns about the reliability of their evaluations. Furthermore, Isotani et al. [23] used a very small dataset of 51 bug reports, with only 5 reports having no duplicates, which limited the robustness of the evaluation.

- **Code Generation**: In the field of code generation, papers Liu et al. [39] and Jana et al. [24] employed similarity-based metrics such as BLEU, CodeBLEU, and Exact Match (EM) for evaluation. While these metrics are useful, they mainly assess syntactic similarity. Liu et al. [39] enhanced the relevance by manually evaluating the functional equivalence of generated code, while Jana et al. [24] introduced Input-Output (IO) equivalence checks, further increasing the relevance and reliability of their evaluation strategies.

- **Manual Code Review**: Thakur et al. [70] used hand-written test benches and manual code reviews to assess the correctness of Verilog code generation. The custom pass@k metric introduced a nuanced way to evaluate performance across different problem categories. However, the lack of documentation about the sampling and review process for manual code review raised minor concerns about the study's reliability.

#### Emerging Key Aspects of Evaluation

From these observations, the following key aspects of evaluation emerged as critical for assessing the reliability and relevance of the reviewed studies:

- **Dataset:** The diversity, size, and representativeness of the dataset directly affect the reliability of an evaluation. Papers using larger, more diverse datasets from real-world or industrial environments typically provide more reliable and relevant insights.

- **Ground Truth:** The quality and construction of the ground truth significantly impact the relevance of the evaluation. For example, efforts to discretize or transform ground truth values (e.g., person-time estimates) to better reflect practical scenarios enhance the relevance of the evaluation.

- **Baseline:** The use of appropriate benchmarks or baselines, such as human expert estimates or well-established machine learning models, adds value to the relevance of the evaluation, especially when the goal is to assess the real-world utility of the LLM-based solutions.

- **Evaluation Metrics:** The choice of metrics used for evaluation—whether accuracy, MAE, BLEU, or custom metrics like IO equivalence—affects both reliability and relevance. Metrics that align closely with the SE task and real-world use cases contribute more meaningfully to assessing a solution's effectiveness.

These aspects form the foundation for evaluating the studies' reliability and relevance in the context of SE tasks and activities. They provide a structured way to assess how well an evaluation method reflects real-world challenges while ensuring that the results are replicable and trustworthy.

### 4.2.4 Refinement of Scope

In the previous phase, **Phase 3** (Section 4.2.3), a subset of papers spanning diverse SE tasks was reviewed to identify key aspects of evaluation that would guide the remainder of this thesis. During this initial review, it became clear that thoroughly analyzing all 295 relevant papers—each focused on evaluating SE solutions—was beyond the practical scope of this thesis. To maintain depth and rigor in the evaluation process, we decided to narrow the focus to **exclude studies** primarily dealing with **code-related tasks**. This refinement aligns with both the practical constraints of the thesis and the research emphasis of DLR, the primary stakeholder, on SE tasks outside of code generation.

**Code-related activities** refer to tasks with a primary focus on code manipulation or generation, including code generation, summarization, review, translation, completion, test case generation, and code repair. Excluding these tasks allowed for a concentrated examination of SE activities focused on process improvement, requirements analysis, and maintenance.

After refining the scope to exclude code-centric papers, 41 papers remained from the original set of 295. The **Software Development** SE activity saw the largest refinement, where only 2 papers remained relevant to the new focus: [82], which explores API documentation augmentation, and [42], which examines requirements completeness. This reduction aligns with the high prevalence of code-centric tasks within Software Development. The **Software Quality Assurance** activity was similarly narrowed, from 52 papers to one paper, while **Software Maintenance** retained a substantial number, with 19 papers remaining from an initial 86, underscoring its relevance within the final review corpus.

Following this refinement, we noted the distribution of the remaining 41 papers by research focus: 38 papers center on the *Development of New SE Solutions*, while 12 focus on the *Performance Assessment of Existing Solutions*. This selection enables a concentrated investigation of evaluation strategies relevant to non-code-centric SE tasks, facilitating a targeted analysis of the reliability and relevance of these methods.

Table 1 illustrates the diversity of SE tasks within each activity area in the refined selection. In particular, Software Design and Development studies emphasize enhancing clarity and completeness, such as specification synthesis and requirements documentation, while Software Maintenance research concentrates on improving process efficiency with tasks like bug prediction, logging, and user feedback analysis. Software Quality Assurance focuses on reliability, addressing fault localization and GUI testing. Research on Software Requirements prioritizes structuring and clarifying requirements through ambiguity detection, traceability, and categorization. Finally, Software Management research centers on optimizing resource planning through tasks like effort estimation. Together, these studies aim to streamline development processes, increase reliability, and enhance clarity across SE activities outside of code

| Software Engineering Activity | Software Engineering Tasks with References |
|---|---|
| Software Design (2) | GUI retrieval (1) [30], Software specification synthesis (1) [47] |
| Software Development (2) | API documentation augment (1) [82], Requirement Completeness Checking (1) [42] |
| Software Maintenance (19) | Bug prediction (1) [15], Bug report related (1) [9], Bug reproduction (1) [22], Bug triage (1) [34], Duplicate Bug Report Detection (1) [23], Stack Overflow Posts summarization (1) [32], API aspect classification (from Stack Overflow Posts) (1) [81],Tag recommendation (Stack Overflow Posts) (1) [18], logging (2) [48, 80], Log parsing (3) [40, 45, 84], Traceability recovery (1) [92], App review feature extraction (1) [52], App Review clustering (1) [77], Sentiment analysis (3) [87, 86, 3] |
| Software Management (3) | Effort Estimation (3) [14, 37, 1] |
| Software Quality Assurance (1) | Mobile app crash detection (1) [41] |
| Software Requirements (13) | Anaphoric Ambiguity Detection (2) [51, 11], Intra-domain Ambiguity detection (1) [50], Coreference detection (1) [76], Requirement analysis and evaluation (2) [56, 60], Requirement traceability (1) [38], Requirements classification (3) [17, 43, 19], Specification formalization (1) [10], Specification generation (2) [44, 78] |
| Software Requirements & Software Design (1) | Use cases generation, System design (1) [85] |

Table 1: Summary of Software Engineering Activities, Tasks, and References

generation.

Following this refinement, Table 2 provides a breakdown of the research focus of the remaining 41 papers, where the majority (30 papers) are concentrated on the *Development of New SE Solutions*, while the remaining 11 focus on the *Performance Assessment of Existing Solutions*. This approach enables a nuanced understanding of the suitability of evaluation strategies used to assess whether the proposed solutions effectively address the distinct needs of SE activities, thereby providing insights for refining evaluation methodologies in future research.

| Research Focus | References |
|---|---|
| Development of New SE Solutions (30) | [47, 44, 76, 9, 10, 42, 32, 23, 50, 56, 38, 34, 43, 19, 48, 80, 77, 40, 45, 84, 22, 52, 18, 37, 92, 82, 14, 51, 30, 41] |
| Performance Assessment of Existing Solutions (11) | [81, 1, 78, 17, 60, 11, 87, 3, 86, 15, 85] |

Table 2: Research Focus Categories with References

By refining the scope to focus on non-code-centric tasks, we are positioned to conduct a more reliable and thorough assessment. This narrowed selection enables an

in-depth review of each evaluation strategy and allows for meaningful discussions with SE experts who bring practical and specialized insights. This collaborative and focused approach strengthens the reliability of the assessment, ensuring a solid foundation for the next stages of the review.

#### 4.2.5 Grouping the papers by underlying task objectives

The goal of this step is to ensure consistency and focus in the review process by grouping papers according to their primary objectives. This organization enables a coherent analysis of evaluation strategies by ensuring that papers addressing similar task objectives are reviewed together, avoiding fragmentation and facilitating a systematic comparison within each group. The thematic organization also highlights the specific challenges addressed and the stakeholders benefiting from these solutions, providing a clear structure for evaluating how well the methods align with the objectives of distinct SE activities.

While some overlap exists across groups due to the interconnected nature of SE tasks, this grouping facilitates a structured and consistent review process. The following categories represent the diverse objectives addressed in the reviewed studies.

**Improving Developer Efficiency - Effort and Resource Estimation**

**Objective:** Enhance developer productivity and project management efficiency through supportive estimations, bug triage prioritization, and knowledge base summarization.

**Stakeholders:** Project managers, developers, and decision-makers.

**Important Evaluation Aspects:** Accuracy in estimations, time savings, usability for developers, and ease of workflow integration.

**Papers:** [1, 14, 37, 23, 34, 32].

**Enhancing Software Reliability and Maintenance**

**Objective:** Improve reliability through automated bug detection, log analysis, and maintenance prediction to enable proactive system upkeep.

**Stakeholders:** Developers, QA teams, and maintenance engineers.

**Important Evaluation Aspects:** Precision and recall, computational efficiency, robustness, and accuracy of localization.

**Papers:** [9, 15, 22, 40, 45, 80, 84, 48, 41].

**User Feedback Processing**

**Objective:** Analyze and structure user feedback from platforms like Stack Overflow and app reviews to support user-driven insights for product development.

**Stakeholders:** Product managers, UX/UI designers, and developers.

**Important Evaluation Aspects:** Classification accuracy, relevance and quality of insights, and adaptability across feedback sources.

**Papers:** [18, 52, 77, 3, 87, 86].

#### Requirements Evaluation and Traceability

**Objective:** Improve requirements documentation by detecting ambiguities, ensuring traceability, and assessing completeness to reduce miscommunication.

**Stakeholders:** Requirements engineers, project managers, and developers.

**Important Evaluation Aspects:** Accuracy in ambiguity detection, traceability precision, completeness validation, and clarity of results.

**Papers:** [17, 19, 60, 43, 11, 50, 51, 76, 42, 38, 56, 92].

#### Program Specifications and API Documentation

**Objective:** Generate and enhance program specifications and API documentation to clarify software functionality and improve developer accessibility.

**Stakeholders:** API consumers, documentation teams, and developers.

**Important Evaluation Aspects:** Specification accuracy, documentation completeness, and support for different levels of abstraction.

**Papers:** [10, 47, 81, 78, 44, 82].

#### Prototyping and System Design

**Objective:** Support early design and prototyping tasks, helping teams rapidly define system requirements and conceptualize solutions.

**Stakeholders:** Designers, product managers, and stakeholders in early product development.

**Important Evaluation Aspects:** Clarity and relevance of prototypes, support for iterative design, and integration with agile workflows.

**Papers:** [30, 85].

### 4.3 Review Process

The stage for the review is set in the previous section, which details the execution of Phases 1 to 3. A focused set of 41 papers, selected from the original set of 396, is identified for non-code-centric SE tasks, ensuring that the evaluation focus in each paper aligns with a solution to a specific SE task. A small but diverse subset of papers is initially reviewed to identify key evaluation aspects, and a scoring system for reliability and relevance is defined. To maintain consistency, the focused set of papers

is grouped by underlying task objectives, enabling papers with similar objectives to be reviewed together.

The objective of this section is to conduct the review in an organized manner, producing concise summaries of the reliability and relevance of the evaluations while extracting quantitative notes and scores on the strengths and limitations of evaluation aspects.

### 4.3.1 Qualitative Review of Evaluation Strategies

The goal of this step was to analyze the evaluation strategies of the selected papers, producing qualitative summaries and scoring their reliability and relevance. The review focused on assessing and summarizing the key evaluation aspects identified in Section 4.2.3.

The author of this thesis conducted the review group-wise, ensuring that papers addressing similar task objectives were analyzed together. For each paper, a summary ranging from 50 to 100 words was written, documenting strengths and limitations in terms of reliability and relevance as defined in Section 4.2.2. Additionally, both dimensions were scored on a scale from 0 to 5.

The review process included analyzing the title, abstract, and introduction to understand each paper's contributions and context. Focus was placed on sections commonly dedicated to datasets (often including descriptions of ground truth creation), metrics, and baselines, as these provided detailed information about the evaluation methodologies. Additionally, tables and figures presenting comparative analyses, benchmarks, and validation methods were examined. Many papers applied cross-validation techniques—such as within-repository, cross-repository [1, 14, 37], or k-fold approaches—or conducted error analyses of predictions [80] and complementary user studies [85]. These observations led to the inclusion of an additional evaluation aspect, *Validation Method*, to systematically document such practices.

For instance, the review for [11] included:

**Qualitative Evaluation Summary Details**

- **Datasets:** DAMIR, ReqEval, CoNLL2011
- **Baselines:** None (standalone evaluation with different configurations)
- **Metrics:** F2-Score, Success Rate
- **Validation Methods:** 10-fold cross-validation, external datasets
- **Ground Truth:** Expert-curated annotations with inter-rater agreement and robust documentation

**Reliability and Relevance Summary:**

- **Reliability:** Robust evaluation using diverse datasets, expert annotations, and metrics addressing class imbalance. Cross-validation and test splits enhanced generalizability. Ground truth creation demonstrated a thoughtful, well-documented process with inter-rater agreement analysis, reinforcing reliability. However, reliance on manual annotation may limit scalability.
- **Relevance:** Emphasis on recall ensures ambiguous cases are flagged for review, addressing critical SE needs. Evaluation on industrial requirements provided strong generalizability, though including open-source datasets could enhance broader applicability. The lack of qualitative studies limited deeper insight into outcomes.
- **Scores:** Reliability: 4, Relevance: 4

#### 4.3.2 Quantitative Analysis of Evaluation Strengths and Limitations

This step focused on systematically extracting strengths and limitations for evaluation aspects from the summaries, enabling quantitative analysis. Each evaluation aspect—datasets, baselines, metrics, validation methods, and ground truth—was scored as follows:

$$\text{Evaluation Aspect's Rating} = \begin{cases} 1 & \text{Strength(s) identified} \\ 0 & \text{No notable comments} \\ -1 & \text{Limitation(s) or mixed outcomes} \\ & \text{(both strengths and limitations identified)} \end{cases}$$

**Evaluation Summary of Key Evaluaiton Aspects**

- **Task:** Ambiguity Detection
- **Reliability:** 4, **Relevance:** 4
- **Datasets: 1** (Diverse datasets: DAMIR, ReqEval, CoNLL2011)
- **Baselines: 1** (Comparison across six paradigms: two SpanBERT-based models, three ML-based models, and one NLP-coreference-based solution)
- **Metrics: 1** (Recall-focused metrics: F2-Score, Success Rate)
- **Validation: 1** (Cross-validation approach)

These structured summaries ensure that both qualitative and quantitative insights are systematically documented for further analysis, laying the foundation for a robust evaluation of current practices.

# 5 Results

## 5.1 Descriptive Summarization of Evaluations by Task Objectives

This section provides a descriptive review of the grouped papers based on their task objectives, with the goal of summarizing the evaluation strategies employed across different task categories. The focus is on systematically describing how the key evaluation aspects identified in **Phase 3**—datasets, baselines, metrics, ground truth, and validation methods—are addressed within each group. By highlighting patterns, strengths, and limitations in these evaluation strategies, this section sets the stage for a critical analysis of the limitations and strengths along key evaluation aspects (5.2) and a insights on reliability and relevance of the reviewed papers evaluaiton strategies(5.3).

The insights provided in this section establish a foundational understanding of the evaluation practices within each task group, ensuring a coherent basis for interpreting the reliability and relevance of the evaluations in later sections.

### 5.1.1 Group 1: Improving Developer Efficiency – Effort and Resource Estimation

The papers in this group aim to enhance developer productivity by addressing tasks such as **effort estimation**, **bug triage and deduplication**, and **knowledge summarization**. Evaluations rely heavily on **quantitative metrics**, with validation strategies like **cross-validation**, **practical alignment with real-world datasets and usage scenarios**, and occasional **user studies**. While datasets predominantly originate from **open-source repositories**, a few papers incorporate **industrial datasets** for increased relevance.

Table 3: Evaluation Setup Summary for Group 1 Papers

| Paper | Datasets | Baselines | Metrics | Validation Methods | Ground Truth |
|---|---|---|---|---|---|
| [1] | DEEP-SE, JOSSE, Porru, PPI, Industrial Dataset | DT with TF-IDF, DT with learned embeddings, Human Expert Estimations | AUC-ROC, F-score | Cross-validation | Actual time spent/ SP |
| [14] | 16 open-source projects | Traditional ML, deep learning, analogy-based methods | MAE | Within- and cross-project scenarios | Estimated SP |
| [37] | JIRA projects, 17 enterprise projects | TF-IDF, LDA, Deep-SE, GPT2SP | MAE, MMRE, PRED(50) | Cross-validation | Actual effort in person-months |
| [23] | NTT Corporation bug reports | TF-IDF, LDA | MAP | 5-fold cross-validation | Human-labeled duplicates and non-duplicates |
| [34] | Google Chromium, Mozilla Core, Mozilla Firefox, Private Industrial Dataset | Transformer-based methods, Traditional ML | acc@k (Top-k Accuracy) | Real-world benchmarking | Actual developers who fixed each bug report |
| [32] | SOSum dataset, additional labeled posts | Heuristic-based, deep learning | Precision, Recall, F1-score | 10-fold cross-validation, user study | Human-labeled summaries |

**Datasets** The datasets used across the group predominantly come from **open-source repositories**, including JIRA project repositories, Mozilla and Google Chromium bug repositories, and the SOSum dataset for knowledge summarization. These datasets offer transparency and reproducibility but often lack domain diversity.

Among the six papers:

- [37] combines both **open-source (JIRA)** and **industrial project datasets**, enhancing real-world generalizability.
- [1] integrates **industrial datasets**, increasing external validity.
- [23] uses a **small proprietary dataset** with only 51 labeled bug reports, limiting statistical robustness.

**Observation:** Open-source datasets dominate, with industrial datasets selectively incorporated. Smaller proprietary datasets risk reducing generalizability.

**Baselines** The baselines used in this group span a range of approaches, reflecting the diverse tasks being addressed:

- **Traditional Machine Learning Methods:** Models such as TF-IDF, LDA, and Decision Trees are used for baseline comparisons, providing foundational benchmarks for evaluating newer methods ([14], [23]).
- **Deep Learning Models:** Transformer-based architectures and neural networks serve as advanced baselines, particularly in tasks requiring contextual understanding and sequence modeling ([34], [14]).
- **Heuristic and Unsupervised Methods:** Task-specific rule-based approaches, heuristic algorithms, and unsupervised techniques provide additional reference points for evaluating task performance ([32]).

The alignment between baselines and research objectives is evident across the studies:

- [14] evaluates their GPT2SP model against traditional ML techniques (e.g., Decision Trees) and deep learning models, ensuring a balanced comparison across baseline types.
- [34] focuses on comparing transformer-based architectures against simpler ML techniques, emphasizing the strengths of transformer models for bug triage tasks.
- [32] incorporates a diverse set of baselines, including a fine-tuned BERT-based summarization model, three heuristic methods, and one unsupervised learning approach, ensuring a multi-faceted evaluation.

**Observation:** Traditional ML methods continue to serve as benchmarks in most tasks, while transformer-based architectures dominate in bug triage and summarization. Kou, Chen, and Zhang [32], highlights the reliance on hybrid evaluation setups that combine heuristic, deep learning, and unsupervised approaches.

**Metrics** Metrics are tailored to task objectives:

- **Effort Estimation:** MAE, MMRE, PRED(50), AUC-ROC ([1], [14], [37]).
- **Bug Triage and Deduplication:** Top-k Accuracy (acc@k), Mean Average Precision (MAP) ([34], [23]).
- **Summarization:** Precision, Recall, and F1-score ([32]).

**Observation:** Metrics align well with task objectives, but qualitative aspects (e.g., user satisfaction) remain underexplored.

**Validation Methods** The predominant validation strategies include:

- **Withing repository and cross-repository validation:** Effort estimation tasks ([1], [14], [37]).
- **Real-World Benchmarking:** Bug triage studies ([34]).
- **User Studies:** Summarization tasks ([32]).

**Observation:** Cross-validation dominates, while user studies remain underutilized, and deployment factors (e.g., latency, scalability) are rarely assessed.

**Ground Truth** Ground truth sources include:

- **Actual Effort Metrics:** Story points, developer time logs ([1]).
- **Estimated Story Points:** Derived from previous estimates ([14]).
- **Manual Annotations:** Bug detection and summarization tasks ([23], [32]).

**Observation:** Actual data improves evaluation relevance, while estimated metrics introduce inherent uncertainty.

**Descriptive Summary** The six papers in this group demonstrate varied evaluation strategies:

- **Datasets:** Open-source datasets dominate; industrial datasets remain underutilized.
- **Baselines:** Traditional ML remains common, but transformers lead in bug triage.
- **Metrics:** Task-specific metrics align well with objectives.
- **Validation:** Cross-validation dominates; user studies are infrequent.
- **Ground Truth:** Actual effort data enhances reliability.

#### 5.1.2 Group 2: Enhancing Software Reliability and Maintenance

The papers in this group aim to improve software reliability and maintenance by addressing tasks such as **bug detection**, **fault localization**, **log analysis**, and **crash reproduction**. Evaluations rely on a combination of **quantitative metrics** (e.g., Precision@K, Balanced Accuracy, F1-score) and **qualitative insights** (e.g., manual validation, ablation studies). Validation strategies prominently feature **cross-validation**, **module-level ablation**, and **manual crash reproducibility checks**. While datasets are primarily derived from **open-source repositories**, a few studies incorporate **industrial datasets** to improve real-world applicability.

Table 4: Evaluation Setup Summary - Bug Detection and Localization

| Paper | Datasets | Baselines | Metrics | Validation | Ground Truth |
|---|---|---|---|---|---|
| [9] | AspectJ, JDT, PDE, SWT, Tomcat, ZXing (OS) | Locus, TBERT-Single, TBERT-Siamese | Precision@K, MAP, MRR | Ablation (granularity, encoding), Runtime analysis | Manually validated mappings (SZZ error mitigated) |
| [15] | Eclipse, Freedesktop, GCC, Gnome, Mozilla, WineHQ (OS, 50K+ reports) | TF-IDF, SVM, RF, k-NN, NB, NN | Balanced Accuracy | 10x5 CV, Wilcoxon test (95%) | Bug fix times (1-year threshold) |
| [41] | 36 widgets, 31 Android apps (OS) | 18 baselines: fuzzing, mutation, string analysis, GUI tools | Detection rate, Attempts, Time | Ablation (modules), GUI integration, Manual validation | Crash reproducibility verified manually and by developers |
| [22] | 75 crash reports, 58 Android apps (OS, IND) | ReCDroid, Monkey, Humanoid, Ape, Q-Testing | Success rate, Time to reproduce | Ablation (scorer impact), Manual validation (14 testers) | Automated reproduction confirmed manually and systematically |

**Datasets** The datasets primarily come from **open-source repositories** like *AspectJ, Eclipse, Mozilla,* and *Android logs*. These datasets provide reproducibility and transparency but often lack the diversity and variability of real-world industrial datasets.

Notable dataset usage patterns include:

- [22] integrates both **open-source crash reports** and **industrial datasets** from Android apps.
- [45] incorporates logs from industrial vendors like **Cisco and Huawei** but lacks .
- [41] relies on a relatively small dataset of **36 widgets from 31 apps**, limiting generalizability.

**Observation:** Open-source datasets dominate, but industrial datasets play a supporting role in enhancing generalizability. Dataset size and diversity remain limiting factors in some studies.

**Baselines** Baseline comparisons include:

- **Traditional IR and ML Approaches:** TF-IDF, SVM, k-NN ([15]).
- **Deep Learning Models:** Transformer-based architectures ([9], [40]).

Table 5: Evaluation Setup Summary - Log Parsing and Analysis

| Paper | Datasets | Baselines | Metrics | Validation | Ground Truth |
|---|---|---|---|---|---|
| [48] | 7,125 Java methods, 1,465 Log4j repos (OS) | Ablation: No Pre-training, Multi-Task, LogStmt-Task, Denoising-Task | Accuracy (level, location), BLEU-4 | Manual validation (300 errors), Ablation studies | Assumed correctness of developer-written logs |
| [84] | 16 Loghub datasets (e.g., HDFS, BGL, Android, Windows) (OS, IND) | Drain, Spell, LenMa, SHISO, Logram, Nulog | Grouping Accuracy, Edit Distance | Ablation (test-time training, variable imitation), Sensitivity analysis | Manually labeled subsets (2K logs/-dataset), minor corrections applied |
| [45] | 96K templates, Cisco, Huawei, H3C logs (OS, IND) | CNN, Bi-LSTM, BERT, RoBERTa, UniLog | Accuracy, Weighted F1, Precision@1, MRR | Low-resource analysis | FPI: Expert-labeled (43 phenomena, multi-label, no conflict details); LDSM: Positive from documentation, negative sampled automatically |
| [40] | (HDFS, BGL, Zookeeper, Android) (OS, IND) | LogPPT, LogStamp, LogParse, LogSig, Spell, Drain, DeepLog, LogAnomaly, LogRobust | F1-score (session-level, template-level), Precision, Recall | Ablation (prompt strategies: Self-prompt, Chain-of-Thought, In-context), Human evaluation (6 experts, 200 logs) | Expert-annotated anomalies and parsing templates, potential pre-training data overlap |
| [80] | 12,012 code snippets from 1,465 GitHub repositories (OS) | LANCE, GPT-3 variants (Ada, Babbage, Curie), Codex (Fine-Tuned) | PA, LA, MA, CLA, CMA, BLEU-4 | Ablation (warmup, example count, example order), Manual error analysis (500 samples) | Log statement correctness validated via annotated dataset and automated metrics |

- **Systematic Testing Tools:** Random testing tools ([22] - Monkey) and crash reproduction frameworks ([22] - ReCDroid).
- **Heuristic Methods:** Rule-based tools for log analysis ([84]).

Examples include:

- [15] evaluates using classical ML approaches like TF-IDF and k-NN.
- [9] employs Transformer-based deep learning models for bug localization tasks.
- [22] utilizes systematic testing tools like *Monkey* (random testing) and *ReCDroid* (crash reproduction).

**Observation:** Baselines generally align with task objectives, with Transformer-based models dominating bug detection tasks, while systematic tools like *Monkey* and *ReCDroid* address crash reproduction. In log parsing tasks, heuristic approaches remain prominent as evaluation baselines, but comparisons with deep learning models (e.g., CNN, LSTM) and large language models (e.g., GPT variants, Codex) are becoming increasingly common.

**Metrics** Metrics are aligned with task objectives, including:

- **Bug Detection and Localization:** Precision@K, MAP, MRR ([9]).
- **Crash Detection:** Detection Rate, Experimental Performance Indicators (e.g., number of attempts, time to trigger crashes) ([41]).
- **Log Analysis:** F1-score, Precision, Recall ([40]), BLEU-4 ([48], [80]), Edit Distance ([84]).

**Observation:** Metrics generally align well with objectives, with traditional quantitative metrics (e.g., Precision@K, MAP, F1-score) dominating bug detection and log analysis tasks. Experimental performance indicators (e.g., number of attempts, time to trigger crashes) play a crucial role in evaluating crash detection scenarios. In log analysis tasks, studies often combine categorical metrics (e.g., position and level accuracy) with text-generation metrics (e.g., BLEU-4, Edit Distance) to assess both the placement and content quality of generated log statements. Despite this, some tasks still lack qualitative validation to complement these quantitative measures.

**Validation Methods** Validation strategies include:

- **Cross-Validation:** Used in bug detection tasks ([15]).
- **Ablation Studies:** Module-level analyses for crash detection ([41]).
- **Manual Verification:** Manual assessment of reproducibility in crash logs ([22]).

**Observation:** Cross-validation is widely used, but deployment-related metrics (e.g., latency, scalability) remain underexplored.

**Ground Truth** Ground truth sources vary across tasks, reflecting the evaluation focus:

- **Manual Verification and Annotation:** Many studies rely on human-annotated datasets, including manually validated mappings for bug localization ([9]) and manually verified crash reproducibility results ([41]). In the case of crash reproduction, validation combines manual checks with automated scripts to ensure consistency and reproducibility ([22]).

- **Historical and System Data:** For long-lived bug prediction, bug fix timelines from issue tracking systems are used as ground truth ([15]).

- **Log Analysis and Generation:** Approaches vary in annotation rigor. [48] uses log datasets extracted directly from software repositories without additional manual validation, while others, such as [80] and [45], incorporate varying degrees of manual annotation. [80] employs detailed validation for log position, verbosity level, and message accuracy, whereas [45] primarily derives ground truth from vendor documentation with limited manual annotations, focusing mainly on Fault Phenomenon Identification (FPI).

**Observation:** Ground truth sources are generally reliable but vary in validation rigor. Manual verification dominates bug detection and crash reproduction tasks, whereas log analysis exhibits a spectrum from fully automated datasets to partially annotated ones.

**Observation:** Ground truth sources are generally reliable but vary in validation rigor. Manual verification dominates bug detection and crash reproduction tasks, whereas log analysis exhibits a spectrum from fully automated datasets to partially annotated ones.

**Descriptive Summary** The papers in this group demonstrate diverse evaluation strategies:

- **Datasets:** Open-source datasets dominate; industrial datasets are selectively used.

- **Baselines:** Classical ML techniques, transformer-based models, and systematic testing tools are most common.

- **Metrics:** Metrics align well with objectives but lack qualitative depth.

- **Validation:** Cross-validation and ablation dominate; deployment factors are underexplored.

- **Ground Truth:** Predominantly manually verified; log analysis ranges from automated datasets to partial manual validation.

### 5.1.3 Group 3: User Feedback Processing

The papers in this group aim to analyze and structure **user feedback** from platforms such as **Stack Overflow**, **app reviews**, and **developer forums**. Tasks addressed include **sentiment analysis**, **tag recommendation**, and **feature extraction**. Evaluations rely primarily on **quantitative metrics**, with validation strategies such as **cross-validation**, **dataset ablation studies**, and **manual validation through expert annotations**. Datasets are predominantly sourced from **platform-specific corpora**, with occasional integration of **crowdsourced annotations** to ensure representativeness.

**Datasets** The datasets in this group primarily focus on **platform-specific user feedback corpora**, sourced from platforms such as:

- **Stack Overflow Posts:** 10.38 million training posts, 100,000 test posts ([18]).
- **App Reviews:** Diverse collections from Google Play Store, Apple Store, and alternative repositories ([77], [52]).
- **Software Engineering Platforms:** Gerrit, Jira, GitHub, and others ([86], [87]).

Annotations vary:

- Manually labeled datasets ([3], [86]).
- Crowdsourced annotations ([52]).
- Mixed manual and automated validation pipelines ([18]).

**Observation:** Dataset quality varies, with stronger annotations derived from inter-rater agreements and iterative refinements, while larger datasets rely more heavily on automated validation.

**Baselines** Baseline comparisons span:

- **Pattern-Based Methods:** SAFE (PoS-based extraction using 18 grammatical patterns), Caspar (dependency parsing with PoS tagging), and KEFE (pattern-based filtering combined with classification) ([77]).
- **Traditional Models:** TF-IDF, LDA, and other statistical approaches ([77]).
- **Deep Learning Models:** RNN, BERT, and Transformer-based methods ([3], [86]).
- **Hybrid Models:** Multi-model combinations, such as PTM4Tag, which blend transformer models with task-specific optimization ([18]).

**Observation:** Transformer-based models dominate sentiment analysis and tagging tasks, while pattern-based and statistical models are primarily used for feature extraction and clustering.

Table 6: Evaluation Setup Summary - Sentiment Analysis & User Feedback Processing

| Paper | Datasets | Baselines | Metrics & Validation | Ground Truth |
|---|---|---|---|---|
| [3] | 5500 SO sentences | RNN4SentiSE | Precision, Recall, F-Measure (10-fold CV, Kappa: 0.88) | Manual, refined guidelines, cross-validation |
| [86] | API reviews, SO posts, app reviews, GitHub, Jira, CR (15K total) | CoreNLP, SentiStrength(-SE), SentiCR, Senti4SD | Precision, Recall, F1, Training Time (70/30 split, cross-dataset) | Manually labeled, SE-specific polarity |
| [87] | SE datasets: Gerrit, GitHub, GooglePlay, Jira, StackOverflow (11K total) | sLLMs (BERT, RoBERTa), bLLMs (Llama2, Vicuna) | Precision, Recall, F1, AUC (80/10/10 split, Wilcoxon test) | Pre-labeled, multiple annotators, conflict resolution |
| [77] | App reviews: 6 apps (3.4K labeled), 18 apps (318K unlabeled) | SAFE, Caspar, KEFE, BiLSTM-CRF, LDA, K-Means | Precision, Recall, F1 (extraction), ARI, NMI (10-fold CV, user survey) | Manual, iterative refinement (Kappa: 0.78–0.86) |
| [18] | Stack Overflow posts (10.38M train, 100K test) | Post2Vec, PTM4Tag (BERT, RoBERTa, CodeBERT, ALBERT, BERTOverflow) | Precision@k, Recall@k, F1@k (Fixed split, ablation, error analysis) | User-labeled tags, filtered rare/noisy tags |
| [52] | App reviews: 468 apps (23.8K reviews, 32.4K annotated features) | SAFE (baseline), BERT, RoBERTa, XLNet | Precision, Recall, F1 (token & feature-level), Precision@k, Recall@k (10-fold CV, lexical overlap analysis, ablation, human validation) | Crowdsourced (AlternativeTo), validated by human annotators (F1: 0.719) |

**Metrics** Metrics are task-specific:

- **Sentiment Analysis:** Precision, Recall, F1-score ([3], [86]).
- **Tag Recommendation:** Precision@k, Recall@k ([18]).
- **Feature Extraction:** ARI, NMI, Lexical Overlap ([77], [52]).

**Observation:** Metrics are well-aligned with task goals, though interpretability of results (e.g., qualitative evaluation) is occasionally overlooked.

**Validation Methods** Validation strategies include:

- **Cross-Validation:** Applied extensively ([3], [77]).
- **Ablation Studies:** To assess model robustness ([18]).
- **Manual Validation:** Expert annotations ([87]).

**Observation:** Cross-validation is the most common strategy, while ablation studies provide additional robustness checks.

**Ground Truth** Ground truth varies:

- **Manual Annotations:** Expert-labeled ground truth ([87]).
- **Crowdsourced Data:** From platforms like AlternativeTo ([52]).
- **User-Generated Tags:** Validated systematically ([18]).

**Observation:** Annotation processes differ, affecting consistency and reliability across studies.

**Descriptive Summary** The papers in this group demonstrate:

- **Datasets:** Platform-specific datasets dominate, with varying annotation quality.
- **Baselines:** Transformer-based models dominate, supported by traditional ML baselines.
- **Metrics:** Task-specific metrics are well-suited but lack interpretability aspects.
- **Validation:** Cross-validation and ablation studies ensure robustness.
- **Ground Truth:** Manual and crowdsourced annotations are common.

#### 5.1.4 Group 4: Requirements Evaluation and Traceability

The papers in this group address critical tasks in requirements engineering, including **requirements classification**, **ambiguity and coreference detection**, **user story evaluation**, and **requirements traceability and completeness**. Evaluations primarily rely on **quantitative metrics** such as F1-score, MAP, and precision to assess the effectiveness of proposed methods. Validation approaches often include **cross-validation**, **cross-dataset testing**, or **manual expert evaluations** to ensure robustness and generalizability. However, some studies are limited by reliance on **qualitative evaluations** conducted on **small, manually reviewed samples**, reducing the scalability and reproducibility of their findings.

Datasets vary across tasks, with some studies relying on well-established benchmarks such as **PROMISE NFR** or **industrial datasets**, while others use **domain-specific corpora** or **synthetic data**. Despite this variety, reliance on older benchmarks and narrowly scoped datasets restricts the broader applicability of the results, particularly in addressing contemporary requirements engineering challenges.

Table 7: Evaluation Setup Summary for Requirements Evaluation Papers

| Paper | Datasets | Baselines | Metrics | Validation | Ground Truth |
|---|---|---|---|---|---|
| [19] | PROMISE NFR (orig/relabeled) | Tree, SVM, Naïve Bayes, CNN | Precision, Recall, F1 | 10-fold CV, loPo | Human-annotated (Functional, NFR subclasses) |
| [43] | PROMISE, NFR-Review, NFR-SO | NoRBERT, BERT-MLM | Precision, F1, T-test | 10-fold CV, trials | Human-annotated labels |
| [17] | PROMISE, Dronology, ReqView | SVM, LSTM, GPT-3.5, GPT-4 | $F_\beta$-Score | Cross-dataset testing | Human-labeled datasets |
| [11] | DAMIR, ReqEval, CoNLL2011 | None | F2, Success Rate | 10-fold CV, external datasets | Expert-annotated labels |
| [50] | Domain-Specific, Multi-Domain Corpus | None | Manual validation | Qualitative analysis | Manual inspection |
| [51] | CS Corpus, PURE Dataset | None | Manual validation | Qualitative analysis | No formal labels |
| [76] | Industry Dataset (21 projects) | None | Precision, F1 | 10-fold CV | Human-annotated labels |
| [60] | Open-source User Stories | AQUSA | Agreement, Precision, F1 | One-shot trials | Double-blinded human eval |
| [38] | Flask, Pgcli, Keras | VSM, LDA, TraceNN | F1, F2, MAP@3 | 10-fold CV, ONS | Mined links (commit tags) |
| [92] | Flask, Pgcli, Keras | TRACEFUN | F1, F2, MAP | 5-fold CV | Inherited mined links |
| [56] | Five hierarchical datasets | IR-based chunking | F2, MAP | 5-fold CV, repeated trials | Manually curated RTMs |
| [42] | PURE Dataset (40 specs) | TF-IDF, WordNet Synonyms | Accuracy, Precision, Coverage | 5-fold CV, human eval | Simulated incompleteness |

**Datasets** The datasets used across this group exhibit a mix of **legacy benchmarks**, **industrial datasets**, and **manually annotated corpora**. PROMISE NFR and its variants remain heavily used in classification tasks, while more domain-specific corpora (e.g., industrial datasets from partner organizations) support ambiguity detection and traceability tasks.

Among the key observations:

- [19], [43], and [17] rely on **PROMISE NFR**, a benchmark dataset for requirements classification, but its age raises concerns about relevance to modern software projects.
- [76] and [42] leverage **industrial datasets**, enhancing real-world applicability.
- [50] employs a **domain-specific Wikipedia corpus**, reducing alignment with real-world requirements engineering documents.

**Observation:** Dataset diversity varies across tasks, with older benchmarks dominating classification tasks and industrial datasets supporting traceability tasks.

**Baselines** Baseline approaches vary significantly depending on the task:

- **Machine Learning Classifiers:** SVM, CNN, Naïve Bayes ([19], [17]).
- **Transformer Models:** BERT, GPT-based models ([43], [60]).
- **Heuristic/Rule-Based Methods:** Used in limited settings ([11]).

Observations include:

- Transformer-based approaches dominate classification tasks ([43]).
- Traditional ML methods remain common in ambiguity detection tasks ([11]).

**Observation:** While ML baselines are well-documented, ambiguity detection tasks lack comparative baselines.

**Metrics** Evaluation metrics are tailored to task-specific objectives:

- **Classification Tasks:** Precision, Recall, Weighted F1 ([19], [43]).
- **Ambiguity Detection:** F2-Score, qualitative validation ([11], [51]).
- **Traceability Tasks:** MAP, Map@k, F2, MRR ([38]).

**Observation:** Metrics are generally aligned with task goals but lack qualitative measures for real-world deployment.

**Validation Methods** Validation approaches include:

- **Cross-Validation:** Stratified and repeated trials ([19], [43]).
- **Human Evaluation:** Manual verification, expert annotations ([42]).
- **Cross-Dataset Testing:** Ensures robustness across datasets ([17]).

Observations:

- Cross-validation dominates classification tasks.
- Ambiguity detection tasks rely heavily on manual validation.

**Observation:** Cross-validation is robust, but reliance on manual validation reduces scalability.

**Ground Truth** Ground truth in this group is primarily derived from:

- **Annotated Datasets:** Datasets labeled by human annotators, often following predefined guidelines ([19], [17], [76]).
- **Manual Inspection:** In some studies, ground truth is validated through focused manual reviews on smaller sample sets ([51]).
- **Absence of Ground Truth (Unsupervised Tasks):** In tasks without labeled data, evaluations rely on qualitative analysis or indirect proxies for correctness, such as clustering consistency or partial alignment with expected outcomes ([50], [51]).

**Observation:** While annotated datasets are widely used across the group, some studies rely heavily on manual validation, limiting scalability and reproducibility. Additionally, unsupervised tasks often lack explicit ground truth, leading to reliance on proxy measures or qualitative assessments, which reduces the robustness of their evaluation outcomes.

**Descriptive Summary** The papers in this group reveal:

- **Datasets:** PROMISE NFR dominates; industrial datasets improve real-world relevance.
- **Baselines:** Transformer-based models excel in classification tasks; ambiguity detection lacks strong baselines.
- **Metrics:** Metrics are task-aligned but lack deployment-focused measures.
- **Validation:** Cross-validation dominates; ambiguity tasks rely on manual validation.
- **Ground Truth:** Annotated datasets are common, but legacy benchmarks reduce relevance.

#### 5.1.5 Group 5: Program Specifications and API Documentation

The papers in this group address challenges related to **program specification generation** and **API documentation augmentation**. These tasks aim to improve **code clarity**, **documentation completeness**, and **developer accessibility** to software components. Evaluation strategies across the group predominantly focus on **quantitative metrics**, supplemented by occasional **qualitative analyses**. Validation approaches include **cross-validation**, **manual error analysis**, and in some cases, **user studies**. While datasets are typically derived from **open-source sources**, some studies rely on **synthetic datasets** to increase data scale ([47]), while others employ **domain-specific benchmarks** such as *SV-COMP*, *SpecGenBench*, or *EvalPlus* to ensure standardized evaluation across tasks ([44], [10]).

Table 8: Evaluation Setup Summary for Program Specifications and API Documentation

| Paper | Datasets | Baselines | Metrics | Validation | Ground Truth |
|---|---|---|---|---|---|
| [81] | 4,522 Stack Overflow API sentences | Opiner, BERT-family, CostSens-BERT | Weighted P/R/F1, MCC, AUC | 10-fold CV, manual analysis | Manual labels (substantial agreement) |
| [82] | APISumBench (4,344 sentences, 48 summaries) | SISE, DeepTip, LexRank, TechSumBot++ | P/R/F1 for classification, ROUGE for summarization | Cross-/Within-API splits, Human review (Likert ratings) | Labeled sentences, Extractive summaries |
| [47] | SpecSyn (300 real specs, 3000 synthetic) | PracExtractor | Precision, Recall, F1 Score | By software types and categories | Manually curated specs from diverse sources |
| [78] | Jdoctor (854), DocTer (2,876) | Jdoctor, DocTer | Accuracy, Precision, Recall, F1 Score | Leave-One-Out CV, Failure Analysis | Annotated datasets, corrected for semantic equivalence |
| [44] | SV-COMP (265 Java); SpecGen-Bench (120); Defects4J (50 files) | Houdini, Daikon, AutoSpec, LLM | Verifier Calls, Passes, Success Rate, User Rating | Comparative study, User survey | Expert specs; Verification-guided benchmarks |
| [10] | EvalPlus (164 Python problems); Defects4J (525 Java bugs) | TOGA, Daikon, GPT-family, StarChat | Accept@k, Bug-Completeness, Qualitative Analysis | Comparative study, Manual analysis | Expert-verified annotations, Test suite with code mutants |

**Datasets** The datasets employed in this group encompass a mix of **open-source repositories**, **evaluation-specific datasets**, and **domain-specific benchmarks**. These datasets vary in size, transparency, and representativeness.

Key observations include:

- [81] uses **4,522 Stack Overflow API sentences**, annotated with substantial inter-annotator agreement, ensuring consistency in labeling.

- [47] evaluates using a **small test set of 250 samples**, with limited transparency regarding the composition of specification (spec) and non-specification (non-spec) examples. The paper suggests a significant class imbalance, implying that the 250 samples likely consist entirely of spec examples. Additionally, an unspecified number of synthetic non-spec examples are introduced for evaluation, but the creation process and distribution criteria for these non-spec samples remain undocumented.
- [44] combines **domain-specific benchmarks** such as *SV-COMP*, *SpecGenBench*, and *Defects4J*, ensuring task diversity and alignment with verification tools.
- [10] employs **expert-verified datasets** and curated benchmarks like *EvalPlus* and *Defects4J*, ensuring reliability in ground truth annotations.

**Observation:** While curated benchmarks like *SV-COMP* and *Defects4J* provide task diversity and align well with verification tools, the evaluation dataset in [47] is limited by a **small sample size** and **lack of transparency** regarding the distribution and creation process of the non-specification (non-spec) examples.

**Baselines** Baseline strategies vary across tasks but generally include:

- **Pre-Trained Transformers:** GPT-family models, BERT-family variants.
- **Dynamic Analysis Tools:** Daikon (for dynamic invariant detection).
- **Annotation Inference Tools:** Houdini (for automatic generation of verification annotations).
- **Graph-Based and ML Models:** LexRank (graph-based summarization), TechSumBot++ (ML-based summarization).

Examples include:

- [81] compares pre-trained transformers against Opiner and CostSensBERT.
- [44] evaluates performance using benchmarks like Houdini, Daikon, and AutoSpec.
- [82] employs LexRank and TechSumBot++ as baseline summarization tools.

**Observation:** Baseline comparisons for **API documentation tasks** frequently involve **Transformer-based models**, while evaluations for **program specification tasks** rely on well-established tools such as **Daikon** for invariant detection and **Houdini** for annotation inference. Additionally, **graph-based (LexRank)** and **ML-based (TechSumBot++)** models serve as comparative benchmarks for summarization tasks.

**Metrics** Evaluation metrics are highly task-specific:

- **API Documentation:** Weighted Precision, Recall, F1, ROUGE, AUC.
- **Program Specifications:** Accuracy, Bug-Completeness, Accept@k.

- **Validation Success Metrics:** Verifier Calls, Pass Rates, User Ratings.

Examples:

- [82] emphasizes ROUGE scores for summarization.
- [10] uses Accept@k and Bug-Completeness for formal verification tasks.

**Observation:** API documentation heavily relies on classification and summarization metrics, while program specification tasks focus on correctness and verification success.

**Validation Methods** Validation approaches include:

- **Cross-Validation:** Standard 10-fold validation.
- **Manual Error Analysis:** Used in almost all papers.
- **User Studies:** Limited but included in tasks like specification generation ([44]).

Examples:

- [82] combines cross-validation with human annotation.
- [44] employs a user survey for qualitative validation.

**Observation:** Cross-validation dominates, while manual error analysis is consistently employed for fine-grained insights.

**Ground Truth** Ground truth sources include:

- **Human Annotations:** Verified annotations for API sentences and specifications.
- **Benchmark Data:** Verification-guided benchmarks like SV-COMP.
- **Synthetic Annotations:** Common in tasks reliant on synthetic specifications ([47]).

**Observation:** Expert-verified datasets strengthen reliability, but reliance on synthetic ground truth introduces limitations.

**Descriptive Summary** The papers in this group demonstrate diverse evaluation strategies:

- **Datasets:** Synthetic datasets reduce real-world applicability.
- **Baselines:** Transformer-based models dominate API tasks; traditional tools are key in formal specification tasks.
- **Metrics:** Metrics are well-aligned with task goals.
- **Validation:** Cross-validation and manual error analysis dominate.
- **Ground Truth:** Benchmark datasets enhance evaluation robustness, but synthetic data reduces generalizability.

### 5.1.6 Group 6: Prototyping and System Design

The two papers in this group focus on supporting early software design and prototyping through tasks such as **interactive programming frameworks** and **natural-language-based GUI retrieval**. Evaluations rely on **task-specific quantitative metrics** for correctness and efficiency, combined with **user studies** for practical validation. The datasets include structured benchmarks and crowdsourced annotations, ensuring both reproducibility and alignment with real-world scenarios.

Table 9: Evaluation Setup Summary - Prototyping and System Design

| Paper | Datasets | Baselines | Metrics | Validation | Ground Truth |
|---|---|---|---|---|---|
| [85] | CAASD (72 software tasks, avg. 240 LoC/-task) | ChatDev, MetaGPT | Pass Rate (%), Token Cost | Manual testing, automatic testing | Reference use cases for functional requirements |
| [30] | Rico GUI dataset (57,764 GUIs), Crowdsourced Gold Standard (100 NL queries, top-20 GUIs/-query) | TF-IDF, BM25, nBOW, PRF-KLD, Sentence-BERT | Precision@k, NDCG@k, MRR, HITS@k, Avg. Precision | Crowdsourced query creation, relevance annotations (AMT), user study (19 participants, real-world tasks) | GUI relevance annotations (3 independent workers/query, majority vote, filtered for quality) |

**Datasets** Zhang et al. [85] introduces the **CAASD benchmark**, a set of 72 tasks averaging 240 lines of code per task, simulating functional programming requirements. However, the origins of these tasks and diversity across domains are not well-documented. Kolthoff, Bartelt, and Ponzetto [30] uses the **Rico GUI dataset** with 57,764 GUIs, enriched by **crowdsourced annotations** for 100 natural-language queries. These annotations, validated by three independent annotators per query, ensure high reliability and coverage. **Observation:** While Zhang et al. [85] focus on structured programming tasks, Kolthoff, Bartelt, and Ponzetto [30] emphasize GUI relevance annotations validated through robust crowdsourced processes. Zhang et al. [85] compares their approach against:**ChatDev** and **MetaGPT** for GUI-Retrieval efficiency from Natural Language Query. **Mockplus** in a user study to evaluate usefulness and productivity improvements for GUI prototyping tasks.

Kolthoff, Bartelt, and Ponzetto [30] evaluates retrieval quality against traditional ranking models like **TF-IDF** and **BM25**, as well as transformer-based models such as **Sentence-BERT**.

**Observation:** Zhang et al. [85] incorporate a two-tier comparison (quantitative baselines and a qualitative user study with Mockplus), while Kolthoff, Bartelt, and Ponzetto [30] focus on retrieval and ranking baselines.

**Metrics** Zhang et al. [85] uses:

- **Pass Rate (%)** to measure task correctness.
- **Token Cost** to assess computational efficiency.

[30] employs:

- Ranking metrics such as **Precision@k**, **NDCG@k**, **MRR**.
- Usability-focused metrics like **HITS@k** and **Average Precision**.

**Observation:** Zhang et al. [85] emphasize correctness and efficiency, while Kolthoff, Bartelt, and Ponzetto [30] focus on ranking accuracy and user-centric relevance.

**Validation Methods** [85] uses:

- **Manual and automatic testing** across CAASD tasks.
- A controlled **user study** comparing their approach against Mockplus, with 19 participants performing prototyping tasks under randomized conditions to reduce bias.

Kolthoff, Bartelt, and Ponzetto [30] integrates:

- A structured **user study with 19 participants**.
- **Crowdsourced query annotations** validated by three independent annotators.

**Observation:** Both studies integrate structured user studies, with Zhang et al. [85] emphasizing productivity comparisons via Mockplus, while Kolthoff, Bartelt, and Ponzetto [30] focus on usability and retrieval relevance.

**Ground Truth** Zhang et al. [85] defines ground truth using **reference use cases for functional requirements**, serving as benchmarks for correctness validation.

Kolthoff, Bartelt, and Ponzetto [30] employs **crowdsourced relevance annotations**, validated by majority voting among annotators and further refined for quality.

**Observation:** Zhang et al. [85] rely on functional correctness benchmarks, while Kolthoff, Bartelt, and Ponzetto [30] use crowdsourced, multi-annotator relevance judgments.

**Descriptive Summary** The two papers in this group demonstrate focused and task-specific evaluation strategies:

- **Datasets:** Structured functional task benchmarks (CAASD) and crowdsourced GUI datasets (Rico).
- **Baselines:** Quantitative frameworks (ChatDev, MetaGPT) and qualitative user study comparisons (Mockplus) for Zhang et al. [85]; retrieval and ranking models (TF-IDF, Sentence-BERT) for Kolthoff, Bartelt, and Ponzetto [30].
- **Metrics:** Task correctness and efficiency (Pass Rate, Token Cost) for Zhang et al. [85]; retrieval accuracy and usability (Precision@k, NDCG@k) for Kolthoff, Bartelt, and Ponzetto [30].
- **Validation:** Controlled user studies (Mockplus comparison in Zhang et al. [85]) and structured crowdsourced query validation [30].
- **Ground Truth:** Functional correctness benchmarks (Zhang) versus multi-annotator validated relevance annotations [30].

## 5.2 Critical Analysis of Strengths and Limitations of Evaluation Strategies

The analysis of evaluation strategies employed in LLM-based Software Engineering (SE) research reveals recurring patterns and limitations across datasets, ground truth, metrics, baselines, and validation methods, examined through the lenses of task objective groups (5.2.1), research focus (5.2.2), ML task types (5.2.3), and temporal trends (5.2.4). Furthermore, a quantitative assessment of overall reliability and relevance scores (**??**) highlights discernible patterns, distinct clusters, and outlier studies in evaluation practices 5.3.2.

Dataset limitations were the most frequently observed, with concerns about representativeness, generalizability, and completeness emerging across multiple task groups. Reliance on narrowly scoped, outdated, or synthetic datasets was a recurring issue. Ground truth construction was often affected by unclear annotation processes and reliance on proxy measures, such as estimated story points in regression tasks, which reduced confidence in evaluation outcomes. While many studies included baseline comparisons, these were frequently absent, simplistic, or non-competitive, with cross-dataset and cross-model comparisons underutilized.

Evaluation metrics varied widely, with some studies using generic or proxy measures that lacked alignment with task objectives. For example, in ranking tasks like requirements traceability, reliance on metrics such as MAP without cutoff reduced practical insights. In contrast, studies employing task-specific, class-balanced, and interpretative metrics demonstrated higher reliability and relevance. Validation strategies were similarly inconsistent, ranging from structured methods like cross-validation, user studies, and ablation tests to more qualitative approaches with limited sample sizes, which reduced reproducibility.

Differences across research focuses (4.2.1) revealed that Performance Assessment of Existing Solutions (PES) studies generally scored higher in both reliability and relevance due to structured validation, robust baseline comparisons, and task-specific metrics. In contrast, Development of New SE Solutions (NSE) studies, while demonstrating strengths in validation diversity, faced recurring challenges in dataset representativeness and baseline design.

### 5.2.1 Evaluation Patterns Across Task Objective Groups

This subsection presents a structured analysis of evaluation strategies across task objective groups, focusing on datasets, ground truth, baselines, metrics, and validation methods. The analysis identifies common limitations and recurring themes across these evaluation aspects, offering insights into the reliability (**RQ1**) and relevance (**RQ2**) of the evaluation strategies in effectively assessing real-world software engineering scenarios. Key themes that emerged during the analysis are summarized across all evaluation aspects, with notable emphasis on dataset **representativeness** (statistical and temporal) and **generalizability** as critical concerns. The findings aim to address **RQ3** by highlighting significant gaps and recurring patterns across evaluation practices.

**Datasets** form the foundation for evaluating the performance and applicability of LLM-based software engineering solutions. The analysis of dataset limitations, as summarized in Table 10, highlights recurring concerns across multiple tasks. Common issues include limitations in dataset size, domain diversity, and clarity in dataset origins or validation processes. These limitations impact the ability of evaluations to generalize findings or accurately reflect real-world software engineering scenarios.

Upon closer inspection, these dataset limitations align with four overarching themes: **Generalizability**, **Representativeness**, **Completeness**, and **Transparency**—as summarized in Table 11. Representativeness can be further divided into two key dimensions: **statistical representativeness**, which refers to dataset size and diversity, and **temporal representativeness**, which concerns whether datasets reflect current software engineering practices or rely on outdated data. These themes encapsulate the key factors that influence the robustness and relevance of dataset choices in evaluation strategies.

These themes represent systematic challenges in dataset design and usage, with implications for the reproducibility and applicability of evaluation outcomes.

**Frequency and Distribution Insights:** The analysis reveals that **representativeness** and **generalizability** are the most frequently observed limitations, each appearing in six papers and spanning diverse tasks such as Bug Detection, Requirements Classification, and Sentiment Analysis. Within representativeness, **statistical representativeness** issues, including dataset size and diversity constraints, are prevalent across Bug Triage and Program Specifications. Conversely,

Table 10: Identified Limitations in Datasets Across Tasks

| Task | Ratio | Observed Limitations |
|---|---|---|
| Requirements Traceability and Completeness | (4/4) | Limited to open-source Python projects; potential incompleteness of mined traceability links [38, 92]; Limited dataset size [56]; Limitation in dataset representativeness for task [42] |
| Automated Requirements Evaluation | (1/1) | Limited dataset size [60] |
| Requirements Elicitation and System Design | (1/1) | Unclear task origins and reference use case creation process [85] |
| Bug Detection and Localization | (3/4) | Limited to open-source projects [9]; Limited widget sample size [41]; Limited crash report sample [22] |
| Ambiguity Detection | (2/4) | Non-requirements corpus [50, 51] |
| Bug Triage and Assignment | (1/2) | Small dataset (51 reports) with bias [23] |
| Requirements Classification | (1/3) | Limited to (old) PROMISE dataset [19] |
| Sentiment Analysis | (1/3) | Limited to SO posts only [3] |
| Program Specifications | (1/4) | Small test set (250 samples) with unclear composition; lack of transparency on whether samples included non-spec data and how these were created [47] |

**temporal representativeness** concerns, such as outdated datasets, are notable in Requirements Classification ([19]).

In contrast, **completeness** and **transparency** are less frequent but still critical. Transparency limitations often arise from ambiguous dataset origins or undocumented annotation processes, particularly in Ambiguity Detection and Requirements Elicitation tasks. Requirements Traceability tasks exhibit limitations across multiple themes, underscoring the inherent complexity of constructing reliable datasets in this domain.

This distribution highlights the need for improved dataset design practices, including broader statistical and temporal coverage, better documentation of annotation processes, and more diverse dataset sources. Addressing these areas will enhance both the reliability and relevance of evaluations in LLM-based software engineering research.

**Ground Truth** serves as the benchmark for evaluating solution performance, with limitations outlined in Table 12. Common issues include **uncertainty in annotation processes**, **unvalidated assumptions**, and **reliance on simulated or artificial ground truths**. Annotation transparency and validation rigor are particularly important for ensuring reproducibility and reliability in ground-truth creation.

Ground truth limitations are less uniformly distributed across tasks but remain

Table 11: Recurring Themes in Dataset Limitations Across Tasks

| Theme | Papers (n) | Associated Tasks | Representative Limitations |
|---|---|---|---|
| **Generalizability** | 6 | Bug Detection, Requirements Traceability, Sentiment Analysis | Limited to open-source datasets [9, 38, 92]; Limited widget sample size [41]; Limited crash report sample [22]; Limited to SO posts [3] |
| **Representativeness** | 6 | Program Specifications, Bug Triage, Requirements Traceability, Ambiguity Detection, Requirements Classification | Small dataset size (51 reports) with bias [23]; Small test set with unclear composition [47]; Simulated incompleteness may not reflect real-world gaps [42]; Wikipedia Corpus not tailored for requirements domain [50, 51]; Outdated dataset (2007 PROMISE dataset) [19] |
| **Completeness** | 2 | Requirements Traceability, Requirements Evaluation | Potential incompleteness of mined traceability links [38, 92] |
| **Transparency** | 2 | Requirements Elicitation, Ambiguity Detection | Unclear task origins and reference use cases [85]; No labeled data validation process [51] |

Table 12: Identified Limitations in Ground Truth Across Tasks

| Task | Ratio | Observed Limitations |
|---|---|---|
| Automated Logging | (1/1) | Assumed correctness without validation [48] |
| Requirements Traceability and Completeness | (2/4) | Potential link incompleteness [38, 92] |
| Effort and Resource Estimation | (1/3) | Ground truth based on uncertain story point estimates [14] |
| Log Parsing and Analysis | (1/4) | Unclear annotation process for some tasks [45] |
| Program Specifications | (1/4) | Nontransparent in class distribution and non-spec creation process [47] |

critical where they occur. In **Requirements Traceability**, the reliance on potentially incomplete mined traceability links emerges as a recurring concern. In **Effort Estimation**, the use of estimated story points as ground truth, rather than actual recorded effort, introduces significant uncertainty and reduces the reliability of the evaluation outcomes. Similarly, tasks such as **Log Parsing and Analysis** and **Program Specifications** exhibit issues related to unclear annotation processes and nontransparent ground-truth creation.

**Baseline** comparisons are crucial for contextualizing the performance of proposed solutions. However, as shown in Table 13, common limitations include the absence of baselines, reliance on limited or non-competitive baseline models, and narrow baseline configurations.

Table 13: Identified Limitations in Baselines Across Tasks

| Task | Ratio | Observed Limitations |
|---|---|---|
| Ambiguity Detection | (3/4) | No baseline comparison [50, 51, 76]; |
| Bug Triage and Assignment | (1/2) | Limited/ Non-competitive baseline selection [23] |
| Program Specifications | (1/4) | Single extractor comparison [47] |

### Metrics

Metrics are essential for quantifying performance. However, as illustrated in Table 14, common issues include reliance on inappropriate metrics, lack of task-specific metrics, and metrics poorly suited for class imbalances.

Table 14: Identified Limitations in Metrics Across Tasks

| Task | Ratio | Observed Limitations |
|---|---|---|
| Automated Logging | (1/1) | Misaligned metrics for log evaluation; poor handling of class imbalance [48] |
| Ambiguity Detection | (2/4) | No cluster quality assessment metrics [50, 51] |
| Bug Triage and Assignment | (1/2) | MAP without cutoff reduces practical ranking insights [23] |
| Program Specifications | (1/4) | Basic classification metrics [47] |
| Requirements Traceability and Completeness | (2/4) | MAP without cutoff reduces practical ranking insights [92] |

### Validation Methods

Validation strategies ensure reproducibility. Table 15 highlights limitations such as reliance on qualitative evaluation and insufficient sample sizes.

Table 15: Identified Limitations in Validation Methods Across Tasks

| Task | Ratio | Observed Limitations |
|---|---|---|
| Ambiguity Detection | (2/4) | Qualitative analysis only with limited sample size [50, 51] |
| Program Specifications | (1/4) | Limited validation approach [47] |

Baseline comparisons, metrics, and validation strategies collectively ensure that evaluation outcomes are interpretable, reproducible, and aligned with real-world objectives. However, recurring limitations are observed across these aspects:

Baselines often suffer from a lack of meaningful baseline comparisons, with limited diversity or reliance on non-competitive baselines (Table 13). In some cases, baseline comparisons are entirely absent, undermining the robustness of performance claims.

Metrics are frequently misaligned with task objectives or insufficient for capturing key performance characteristics. For example, MAP without cutoff reduces practical ranking insights in bug triage and requirements traceability tasks. Similarly, class imbalance in tasks like automated logging is often poorly addressed (Table 14).

Validation practices exhibit limitations in both scope and transparency. Qualitative analyses, while useful for providing nuanced insights, are often conducted with limited sample sizes and lack standardization. Structured validation strategies, such as cross-validation or statistical significance testing, are inconsistently applied across tasks (Table 15).

Additionally, across tasks and evaluation aspects, a notable observation is the inconsistent integration of qualitative evaluation methods, such as error analysis and user studies, which can provide complementary insights to quantitative metrics. For example:

- In Xu et al. [80], a qualitative error analysis was performed on 500 randomly selected log messages, categorizing errors into semantic mismatches, incorrect variable predictions, and meaningless descriptions. This analysis offered actionable guidance for improving LLM-based logging systems but was not systematically replicated across other studies.
- In Yang et al. [81], error analysis identified critical insights into out-of-vocabulary issues and comparative strengths of different approaches using Venn diagrams to visualize overlaps and differences in correct predictions.

Despite their value, such qualitative evaluations remain sporadic and lack consistent adoption across different tasks and studies. Future evaluation strategies should emphasize the integration of structured qualitative error analysis and user studies to complement quantitative results, improving both reliability and relevance.

The results reveal systemic patterns across evaluation strategies, with several recurring challenges affecting both reliability and relevance:

Reliability is frequently undermined by dataset limitations, especially around representativeness and transparency, which affect reproducibility. Similarly, inconsistencies in validation methods, insufficient baseline comparisons, and underutilization of structured qualitative evaluation reduce confidence in reported findings.

Relevance is often restricted by misaligned metrics and datasets with limited generalizability, reducing the ability of evaluations to reflect real-world software

engineering scenarios accurately. The absence of qualitative evaluation practices, such as error analysis or user studies, further limits practical insights.

Addressing these shortcomings requires improvements across multiple dimensions:

- Enhanced dataset design with a focus on representativeness (both statistical and temporal). Clearer documentation and standardization of validation processes. Metrics aligned with task-specific goals, particularly in scenarios with class imbalance or ranking objectives.
- Broader adoption of structured qualitative evaluation methods, such as error analysis and user studies, to complement quantitative results. Rigorous and diverse baseline comparisons to provide meaningful performance benchmarks.

These insights contribute directly to answering RQ3, identifying actionable focus areas for enhancing evaluation strategies in LLM-based SE research.

### 5.2.2 Evaluation Patterns Across Research Focus of the Papers

The evaluation strategies observed across the two primary research focuses—Development of New SE Solutions (NSE) and Performance Assessment of Existing Solutions (PES)—reveal distinct patterns and priorities across key evaluation aspects. While NSE studies demonstrate notable strengths, particularly in validation methodologies, their performance in other aspects shows recurring limitations. In contrast, PES studies exhibit consistently robust evaluation practices, with fewer limitations overall. However, the imbalance in sample sizes—31 NSE studies versus 10 PES studies—must be considered when interpreting these patterns.

Validation emerges as the most prominent strength in NSE studies, with a significant majority employing cross-validation strategies, comparative analysis, ablation studies, user studies, and manual validation approaches (Table 16). Cross-validation methods are extensively applied, often incorporating sensitivity analyses and statistical measures to enhance result reliability. Comparative approaches, including benchmarking across datasets and methods, provide valuable insights into solution performance. Ablation studies are frequently used to isolate and measure the impact of individual components or design choices within the evaluated solutions. User studies further complement these methods by adding a layer of practical insight into solution usability. However, NSE studies occasionally exhibit reliance on qualitative validation methods, limited validation scope, and a lack of robust statistical analysis, resulting in variable reproducibility (Table 18). In contrast, PES studies prioritize controlled and statistically grounded validation approaches. Cross-validation methods in PES studies are typically accompanied by statistical significance testing and comparative benchmarking across diverse datasets (Table 17). This focus on methodological rigor ensures that PES validation results are reproducible and interpretable, albeit with a narrower range of validation strategies compared to NSE studies.

Table 16: Thematic Strengths in Development of New SE Solutions (NSE)

| Evaluation Aspect | Ratio (n/31) | Thematic Strengths |
|---|---|---|
| Validation | 28/31 | **Cross-validation strategies:** [14, 37, 34, 30, 38, 43, 19, 45, 52, 77, 80, 48, 84, 56, 86, 82, 81, 76]. **Comparative analysis:** [10, 32, 44, 92, 42]. **User studies:** [32, 77, 44, 30, 14]. **Ablation studies:** [9, 18, 22, 80, 85, 41, 45, 42]. **Manual validation and error inspection:** [22, 84, 30, 52, 77]. |
| Baseline | 24/31 | **Diverse paradigms:** [14, 37, 32, 34, 40, 43, 80, 77, 92]. **Tool comparisons:** [22, 44, 30, 42, 45, 52, 19, 9]. **Variant comparisons:** [18, 52, 10, 85]. |
| Metrics | 24/31 | **Task-specific metrics:** [14, 32, 80, 77, 82, 85, 19, 10]. **Balanced metrics:** [37, 18, 86, 43, 52, 44, 38, 40]. **Comprehensive metric suites:** [34, 22, 41, 42, 84, 30, 45]. |
| Ground Truth | 21/31 | **Human validation:** [14, 32, 80, 43, 30, 38, 41, 77]. **Expert annotations:** [22, 44, 85, 19, 9, 40]. **Inter-rater agreement:** [84, 52, 56]. **Systematic validation:** [45, 52, 42]. **Real-world assignments and effort validation:** [34, 37]. |
| Datasets | 18/31 | **Dataset diversity:** [14, 32, 37, 34, 42, 85, 9, 10]. **Balanced splits:** [40, 80, 30, 44, 43]. **Well-annotated datasets:** [22, 44, 52, 9, 19]. **Large-scale datasets:** [18, 45, 38]. **Real-world data inclusion:** [22, 37]. |

In terms of baseline comparisons, NSE studies emphasize diverse paradigms, variant comparisons, and tool-based evaluations (Table 16). However, recurring limitations such as single-tool reliance, simplistic baselines, or absence of comparative baselines are evident (Table 18). Conversely, PES studies apply focused comparisons between tools, models, and paradigms, ensuring precise benchmarking (Table 17). While PES studies occasionally exhibit narrower baseline diversity, they demonstrate methodological clarity and relevance in their comparative approaches.

When examining metrics, NSE studies frequently employ task-specific and purpose-oriented metrics tailored to the evaluation objective (Table 16). A recurring strength lies in their appropriate handling of imbalanced datasets, ensuring fairness in performance assessments. However, NSE studies sometimes rely on limited or overly simplistic metrics and occasionally omit quantitative assessments (Table 18). PES studies, in contrast, consistently apply class-balanced metrics, task-specific metrics, and comprehensive suites of performance measures (Table 17). Their metric choices are often aligned with the unique challenges posed by their evaluation scenarios.

For ground truth, NSE studies emphasize manual validation, expert annota-

Table 17: Thematic Strengths in Performance Assessment of Existing Solutions (PES)

| Evaluation Aspect | Ratio (n/10) | Thematic Strengths |
|---|---|---|
| **Validation** | 10/10 | **Cross-validation strategies:** [1, 3, 17, 15, 60, 78, 81, 86, 87, 11]. **Statistical significance testing:** [60, 11]. **Cross-dataset validation:** [17, 87]. |
| **Metrics** | 10/10 | **Class-balanced metrics:** [1, 17, 86, 87]. **Task-specific metrics:** [3, 15, 81]. **Comprehensive metric suites:** [11, 78]. **Addressing class imbalance:** [1, 60]. |
| **Ground Truth** | 10/10 | **High inter-rater agreement:** [1, 3, 86, 87]. **Manual SE-specific annotations:** [17, 87]. **Clear annotation processes:** [60, 81]. **Double-blind human evaluation:** [78]. **Semantic validation:** [81]. **Expert agreement in annotations:** [15]. **Multiple annotators with resolution:** [11]. |
| **Baseline** | 9/10 | **Tool comparisons:** [78, 60, 81]. **ML and human expert baselines:** [1]. **Paradigm diversity:** [17, 15]. **Limited baseline comparison:** [3]. **Model comparisons:** [87, 86]. |
| **Datasets** | 8/10 | **Dataset diversity:** [1, 3, 17]. **Well-annotated datasets:** [78]. **Large-scale datasets:** [15]. **Multiple SE domain datasets:** [81, 86]. **Validated datasets:** [60]. |

tions, inter-rater agreement assessments, and systematic validation processes (Table 16). Despite these strengths, limitations persist, including uncertain annotations, reliance on unvalidated manual labeling, and inherited dataset incompleteness (Table 18). PES studies demonstrate structured annotation processes, high inter-rater agreements, and clear expert validation protocols (Table 17). While these strategies ensure reliability, PES studies remain dependent on pre-existing datasets, which can introduce constraints in representativeness.

The handling of datasets represents one of the most pronounced differences between the two research focuses. NSE studies often emphasize diverse datasets, balanced splits, and large-scale collections (Table 16). However, they frequently face challenges such as limited dataset diversity, reliance on synthetic data, task-specific biases, and small sample sizes (Table 18). PES studies, on the other hand, make use of well-annotated, validated, and domain-representative datasets (Table 17). Despite their methodological consistency in dataset utilization, limitations still surface in terms of dataset size and diversity, as highlighted by the recurring reliance on narrow dataset scopes in two PES studies (Table 19).

Notably, many NSE studies not only evaluate their proposed solutions but also contribute new datasets and resources as secondary outputs. These datasets vary widely in scope and purpose. Some are closely tailored to the specific tasks

Table 18: Thematic Limitations in Development of New SE Solutions (NSE)

| Evaluation Aspect | Ratio (n/31) | Thematic Limitations |
|---|---|---|
| **Validation** | 3/31 | **Limited validation approaches:** [23, 47, 50]. **Reliance on qualitative analysis:** [23, 47]. |
| **Metrics** | 5/31 | **Limited metric diversity:** [23, 47, 50]. **Lack of quantitative metrics:** [51, 76]. **Reliance on basic classification metrics:** [47]. |
| **Ground Truth** | 9/31 | **Uncertain or incomplete annotations:** [23, 50, 51, 76]. **Manual labeling without validation:** [47, 19, 22]. **Inherited incompleteness in ground truth data:** [38, 9]. **Unclear annotation processes:** [51]. |
| **Baseline** | 5/31 | **Limited baseline diversity:** [23, 50]. **Absence of baseline comparisons:** [51, 76]. **Single extractor comparisons:** [47]. |
| **Datasets** | 13/31 | **Limited dataset diversity:** [23, 47, 76, 85]. **Small dataset size:** [23, 51, 9]. **Task-specific limitations:** [56, 38]. **Synthetic data reliance:** [22]. **Domain-specific bias:** [41, 42]. **Limited to non-requirements data:** [38, 51]. **Bias in sampling:** [19]. |

addressed in the study, such as **SOSum (Stack Overflow Summary Dataset)** for summarization tasks [32] and **API documentation datasets** for API analysis [82]. However, other datasets, while large-scale and general in nature, diverge from typical real-world task data. For example, **NFR-SO (Non-Functional Requirements Stack Overflow dataset)** [80] aggregates large-scale recent content from Stack Overflow but differs in structure and focus compared to conventional requirements datasets. Similarly, datasets constructed from Wikipedia corpora for intradomain ambiguity detection [50, 51] introduce significant scale but lack direct alignment with typical software engineering requirements, raising questions about their real-world applicability.

While these contributions undoubtedly enrich the SE evaluation landscape by offering novel data sources, they also highlight recurring challenges. The reliance on self-curated or domain-divergent datasets can introduce biases, task-specific constraints, or domain mismatches, limiting broader generalizability across diverse SE tasks. This dual role of NSE studies—simultaneously proposing solutions and creating evaluation resources—adds a layer of complexity when assessing the overall reliability and representativeness of their evaluation strategies.

In summary, NSE studies excel in validation diversity and methodological breadth, leveraging multiple evaluation approaches to address real-world applicability. However, they frequently encounter limitations in consistency, baseline comparisons, metric diversity, and dataset representativeness. PES studies, while fewer in number, consistently apply methodologically rigorous evaluation strategies

Table 19: Thematic Limitations in Performance Assessment of Existing SE Solutions (ESE)

| **Evaluation Aspect** | **Ratio (n/10)** | **Thematic Limitations** |
|---|---|---|
| Datasets | 2/10 | **Limited dataset diversity:** Reliance on datasets restricted to Stack Overflow (SO) posts [3].<br>**Small dataset samples:** Use of small samples of requirements data, limiting generalizability [60]. |

across multiple aspects. They display strengths in comparative metrics, structured validation, and well-annotated datasets. Nonetheless, recurring dataset limitations suggest potential areas for refinement. These patterns set the stage for deeper reflection and critical discussion in the subsequent section.

### 5.2.3 Influence of ML Task Types on Evaluation Strategies

The comparative analysis of evaluation strategies across different ML task types reveals notable patterns concerning recurring limitations and strengths. While no clear causal relationship between ML task type and evaluation quality can be established, the observed trends highlight key areas where evaluation strategies demonstrate weaknesses or strengths.

**Ambiguity Detection** studies, primarily framed as *classification tasks* [50, 51], face consistent limitations across multiple evaluation aspects. These include the absence of quantitative baselines, reliance on qualitative validation approaches, and the use of datasets that are *not representative* of typical software requirements documents. For example, Wikipedia corpora are used instead of domain-specific datasets, which reduces the applicability of the evaluation results to real-world software engineering tasks. Both studies rely on clustering techniques, which introduce unique evaluation challenges due to the absence of labeled data for clear ground truth comparisons. Despite these inherent challenges, minimal effort is observed in applying established evaluation practices tailored for such unsupervised tasks.

In contrast, **Automated Logging**, addressed through generation tasks [48], exhibits limitations primarily in the lack of efforts to assess the quality of the logs used as data. These study assume the correctness of the logs without reviewing their quality, potentially introducing biases or inaccuracies. Additionally, the reduced generalizability stems from the exclusive use of logs from **Java projects using Log4j**, limiting generalizability to other programming languages or logging frameworks.

**Bug Detection and Localization** displays notable differences between *classification* and *generation approaches*. Studies adopting *classification approaches* [9, 41] often suffer from dataset-related limitations, including reliance on open-source data and small sample sizes. Meanwhile, *generation approaches* [22] encounter

fewer limitations but still show dataset-related weaknesses, particularly concerning the scale and representativeness of the data.

In **Bug Triage and Assignment**, studies using *generation tasks* [23] exhibit more limitations compared to their *classification counterparts*. Observed weaknesses include reliance on manual labeling without quality metrics, limited baseline comparisons, and simplistic evaluation metrics. These limitations reduce the robustness and comparability of results.

For **Effort and Resource Estimation**, *regression-based studies* [14] highlight challenges with uncertain ground truth data, specifically the reliance on estimated rather than empirically measured values (e.g., story points). This limitation raises concerns about the reproducibility and reliability of evaluation results. In contrast, *classification-based studies* display fewer observed limitations in their evaluation practices.

**Requirements Traceability and Completeness** shows recurring weaknesses in studies using *recommendation tasks* [38, 92, 42], where incomplete ground truth data and the limited representativeness of datasets frequently surface as primary concerns. In comparison, *classification-based studies* exhibit fewer limitations, particularly in validation methodologies.

**Program Specifications**, when framed as *classification tasks* [47], show significant limitations, including reliance on synthetic datasets, basic evaluation metrics, and simplistic validation strategies. *Generation-based approaches* demonstrate fewer observed weaknesses, suggesting a stronger alignment between evaluation practices and task objectives.

Tasks such as **Sentiment Analysis** and **Log Parsing and Analysis** consistently display fewer evaluation limitations across both *classification* and *generation framings*. These tasks benefit from established datasets, reproducible metrics, and validation methodologies, contributing to more robust evaluation practices.

These insights are summarized in Table 20, which highlights key evaluation limitations across ML task types and references associated studies.

**Key Insights:** While ML task types cannot be identified as causal factors for evaluation limitations, patterns emerge indicating recurring weaknesses tied to specific tasks. Ambiguity Detection and Bug Triage tasks face substantial evaluation challenges, while Sentiment Analysis and Log Parsing tasks exhibit relatively consistent evaluation robustness. Dataset representativeness and validation methodologies remain persistent concerns across multiple ML-task types.

The observations suggest that dataset diversity, baseline comparisons, and metric selection require more attention, regardless of the ML task framing. Additionally, more structured approaches to handling evaluation in unsupervised or clustering-based tasks may help mitigate some of the observed limitations.

### 5.2.4 Temporal Trends in Evaluation Strategies

The temporal analysis of evaluation strategies reveals recurring and evolving patterns in the limitations observed across studies from 2020 to 2024. While

Table 20: Key Evaluation Limitations Across ML Tasks in Software Engineering

| Task | ML Task | Key Limitations | References |
|---|---|---|---|
| **Ambiguity Detection** | Classification | No quantitative ground truth, no baseline comparison, reliance on qualitative analysis, non-representative datasets | [50, 51, 76] |
| **Automated Logging** | Generation | Assumed correctness without validation, limited metric selection | [48] |
| **Automated Requirements Evaluation** | Classification | Small dataset sample limits generalizability | [60] |
| **Bug Detection and Localization** | Classification | Dataset limitations (open-source only, widget sample size) | [9, 41] |
| **Bug Triage and Assignment** | Generation | Manual labeling without quality metrics, limited baseline comparison, simplistic evaluation metric | [23] |
| **Effort and Resource Estimation** | Regression | Uncertain ground truth (story point estimates) | [14] |
| **Log Parsing and Analysis** | Generation | Unclear annotation processes | [45] |
| **Program Specifications** | Classification | Limited real specifications, basic metrics, reliance on synthetic data | [47] |
| **Requirements Traceability and Completeness** | Recommendation | Incomplete ground truth, non-representative datasets | [38, 92, 42] |
| **Requirements Elicitation and System Design** | Generation | Limited task diversity | [85] |
| **Sentiment Analysis** | Recommendation | Dataset restricted to Stack Overflow posts | [3] |

the number of studies per year fluctuates, some persistent trends emerge, particularly concerning dataset representativeness, validation practices, and task-specific methodological choices.

Table 21: Evaluation Limitations in 2020 - Total 3 Papers

| Evaluation Aspect | Limitations (Cited Papers) |
|---|---|
| **Dataset (2 papers)** | [19]: Limited to PROMISE dataset<br>[3]: Limited to SO posts only |

Table 22: Evaluation Limitations in 2021 - Total 2 Papers

| Evaluation Aspect | Limitations (Cited Papers) |
|---|---|
| **Ground Truth (2 papers)** | [38]: Potential link incompleteness<br>[23]: Manual labeling without quality metrics |
| **Baseline (1 paper)** | [23]: Limited baseline comparison |
| **Metrics (1 paper)** | [23]: Single metric (MAP) without context |
| **Dataset (2 papers)** | [38]: Limited to Python projects<br>[23]: Small dataset (51 reports) with bias |

**Dataset limitations** consistently appear across all years, indicating ongoing challenges in dataset diversity, size, and representativeness (see Tables 21–25). In many cases, studies rely on narrowly scoped datasets, synthetic data, or domain-specific corpora, which reduce the generalizability of evaluation outcomes.

**Ambiguity Detection** studies demonstrate recurring limitations across multiple

Table 23: Evaluation Limitations in 2022 - Total 11 Papers

| Evaluation Aspect | Limitations (Cited Papers) |
|---|---|
| **Ground Truth (3 papers)** | [48]: Assumed correctness without validation<br>[50]: No quantitative ground truth<br>[92]: Inherited potential incompleteness |
| **Baseline (2 papers)** | [50]: No baseline comparison<br>[76]: No baseline comparison |
| **Metrics (2 papers)** | [48]: Limited metric selection<br>[50]: No quantitative metrics |
| **Validation (1 paper)** | [50]: Qualitative analysis only |
| **Dataset (3 papers)** | [9]: Limited to open-source projects<br>[50]: Non-requirements corpus<br>[92]: Limited to specific projects |

years, particularly in aspects of **baseline comparisons** and **metrics selection**. Studies from the same research group across 2022 and 2023 (e.g., [50], [51]) exhibit similar shortcomings, such as the absence of quantitative metrics, reliance on qualitative validation, and a lack of robust baselines.

**Ground truth limitations** evolve over time. Earlier studies (e.g., [23]) focused on manual labeling challenges, while later works (e.g., [14]) highlighted reliance on uncertain proxy measures, such as estimated story points, instead of empirically validated ground truth data.

Table 24: Evaluation Limitations in 2023 - Total 17 Papers

| Evaluation Aspect | Limitations (Cited Papers) |
|---|---|
| **Ground Truth (3 papers)** | [47]: Limited real specifications<br>[51]: No labeled data validation<br>[14]: Ground truth based on uncertain story point estimates |
| **Baseline (2 papers)** | [47]: Single extractor comparison<br>[51]: No baseline comparison |
| **Metrics (2 papers)** | [47]: Basic classification metrics<br>[51]: No quantitative metrics |
| **Validation (2 papers)** | [51]: Qualitative analysis only<br>[47]: Limited validation approach |
| **Dataset (6 papers)** | [42]: Limited specifications sample<br>[51]: Non-requirements corpus<br>[56]: Limited dataset size<br>[41]: Limited widget sample size<br>[60]: Small requirements sample<br>[47]: Predominantly synthetic data |

In summary, while certain evaluation challenges remain persistent, task-specific trends and methodological choices have a significant influence on the presence

Table 25: Evaluation Limitations in 2024 - Total 8 Papers

| Evaluation Aspect | Limitations (Cited Papers) |
|---|---|
| **Ground Truth (1 paper)** | [45]: Unclear annotation process for some tasks |
| **Dataset (2 papers)** | [85]: Limited task diversity<br>[22]: Limited crash report sample |

and severity of these limitations. The tables embedded in this subsection (Tables 21–25) provide a detailed overview of these trends, referencing the specific limitations and associated studies.

## 5.3 Insights on Reliability and Relevance of the studies

The quantitative evaluation of strengths and limitations of the reviewed evaluation practices in the previous (Section 5.2) revealed clusters of high-performing studies with robust datasets and evaluation strategies, as well as outliers characterized by methodological weaknesses or exemplary practices.

In the context of ML task types (5.2.3), no causal relationship was established between task type (e.g., classification, generation, regression) and evaluation quality. However, patterns emerged: classification tasks frequently faced challenges with dataset representativeness and validation transparency, while generation tasks often like for automated logging fall short of reviewing and validating the quality of used test data and the use of metrics captuing all aspects relevant for assign real world applicability. Across all dimensions, no significant temporal trends were observed (5.2.4), with dataset limitations and validation inconsistencies persisting across all years despite modest improvements in task-specific metrics and validation practices.

### 5.3.1 Overal Reliablity and Relevance

The preceding subsections examined evaluation strategies across multiple dimensions, uncovering recurring limitations in *dataset representativeness, baseline diversity and competitiveness, metrics suitability, validation scope, and ground truth clarity*. These limitations affect the overall **reliability** and **relevance** of evaluation outcomes, with discernible patterns emerging across task objective groups, research focuses, and publication years.

Building on these observations, this subsection consolidates the findings, beginning with an overview of the **overall reliability and relevance** scores observed across the reviewed studies. These insights provide a foundation for understanding recurring challenges and areas of strength in current evaluation practices.

**Overall** the evaluation strategies across the reviewed studies reveal a **moderate performance** in both **reliability** and **relevance** dimensions, with notable

limitations affecting their overall effectiveness. For **reliability**, the *mean score of 3.20* and *median score of 3.0* indicate that **most evaluation methodologies achieve only a baseline level of scientific rigor and reproducibility**. While no study scored a *0*, signaling that all evaluations included some methodological effort, recurring weaknesses—such as *insufficient validation, reliance on limited datasets, and lack of methodological transparency*—prevent higher reliability scores.

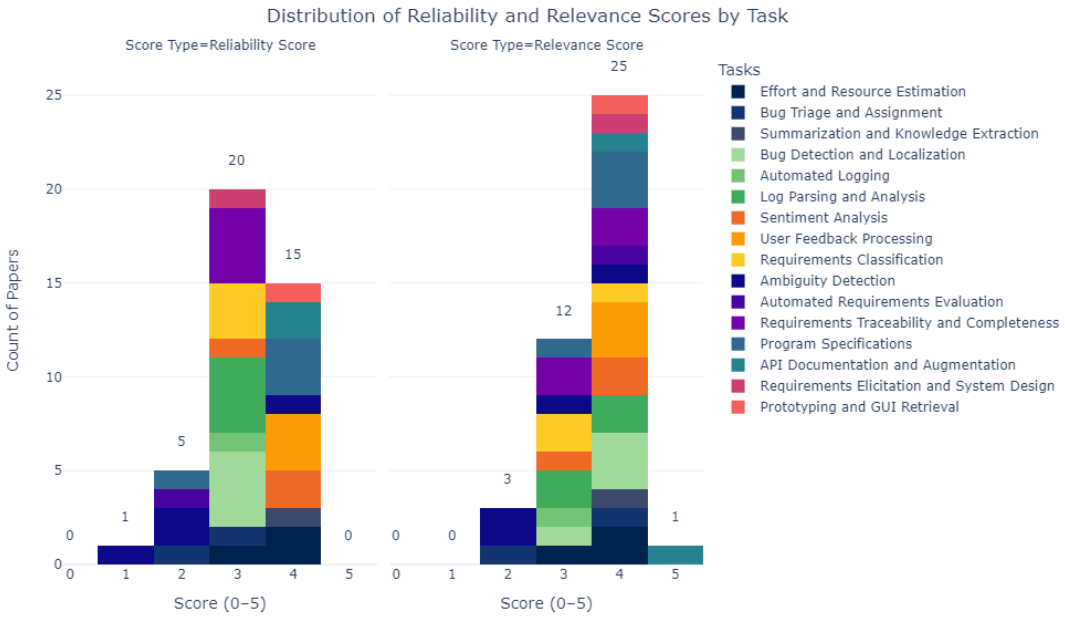



Figure 7: Distribution of reliability and relevance scores across tasks

In contrast, **relevance** scores suggest slightly better alignment with **real-world applicability**, with a *mean score of 3.59* and a *median score of 4.0*. However, this alignment remains inconsistent across tasks, often constrained by *over-reliance on proxy metrics, simplified assumptions, and task-specific biases in dataset design*.

These moderate scores in both dimensions highlight **shortcomings** in evaluation strategies, including *dataset representativeness, baseline robustness, and transparency in validation methods*. While some evaluations demonstrate thoughtful design and alignment with real-world conditions, many remain hindered by methodological weaknesses that undermine their ability to provide **trustworthy and practically meaningful insights**.

### 5.3.2 Summary of Reliability and Relevance Across Dimensions

The analysis reveals distinct patterns in the reliability and relevance of evaluation strategies across **years**, **research focuses**, and **tasks**, offering insights into the strengths and limitations of current practices.

**Over time**, no clear temporal pattern in reliability and relevance scores is evident. While minor improvements are observable in recent years, particularly

in the percentage of studies with limitations related to *datasets*, these trends remain inconclusive. The years **2023** and **2024** show a lower percentage of dataset-related limitations compared to **2020** and **2021**. However, these differences should be interpreted cautiously, as the earlier years have significantly fewer studies (**2020: 3 papers**, **2021: 2 papers**), while the majority of reviewed studies are from **2023** and **2024**. This uneven distribution limits the validity of drawing strong conclusions from temporal trends alone.

When comparing the two primary **research focuses**, notable differences emerge. Evaluations centered on the **Performance Assessment of Existing Solutions (PES)** generally achieve **higher reliability and relevance scores** and display limitations primarily in the *datasets* used for evaluation. In contrast, evaluations focusing on the **Development of New SE Solutions (NSE)** reveal a broader spectrum of limitations, spanning *datasets, baselines, metrics, ground truth*, and *validation methods*. These findings suggest that PES studies benefit from more mature evaluation methodologies, while NSE studies face challenges in establishing robust and reproducible evaluation frameworks.

At the **task level**, clear patterns emerge, forming distinct clusters based on evaluation performance. A group of tasks, including *API Documentation and Augmentation*, *Prototyping and GUI Retrieval*, *Summarization and Knowledge Extraction*, and *User Feedback Processing*, consistently demonstrate **high reliability and relevance**. These tasks are often closely aligned with broader NLP challenges or involve inherently practical evaluation setups, contributing to their strong methodological performance.

In contrast, tasks such as *Ambiguity Detection*, *Bug Report Deduplication*, and *Automated Requirements Evaluation* frequently exhibit **lower reliability and relevance scores**, with recurring limitations across multiple evaluation aspects. These tasks often face challenges in defining clear evaluation metrics, constructing representative datasets, and establishing reliable ground truths, which may stem from the inherent complexity and ambiguity of the tasks themselves.

In summary, while gradual improvements are visible across years and significant differences exist between research focuses, task-level patterns reveal the most pronounced contrasts. These findings set the stage for the cluster analysis that follows, where the interplay between reliability, relevance, and evaluation aspects across distinct groups of studies is examined.

**Cluster Patterns in Evaluation Performance** Figure 8 illustrates the distribution of studies across reliability and relevance scores, highlighting several notable trends. First, the reviewed papers generally exhibit **slightly higher relevance scores compared to reliability scores**, suggesting that while evaluations often align with practical task objectives, they frequently face challenges related to reproducibility and methodological rigor. However, a significant number of studies still fall into moderate and low relevance clusters, indicating inconsistencies in addressing real-world applicability.

Second, a clear trend emerges in the performance of studies based on their

**research focus**. Papers categorized under the **PES** generally achieve higher reliability and relevance scores compared to those under the **NSE** focus.

Third, patterns are also observable across **task objective groups**. For example, tasks in **Group 3: User Feedback Processing**—including *Sentiment Analysis* and *User Feedback Processing*—tend to achieve consistently high reliability and relevance scores. Similarly, tasks in **Group 2: Enhancing Software Reliability and Maintenance**—such as *Bug Detection and Localization*, *Automated Logging*, and *Log Parsing and Analysis*—demonstrate moderate to high performance along relevance axis and only moderate reliability, indicating task-specific patterns in evaluation outcomes.

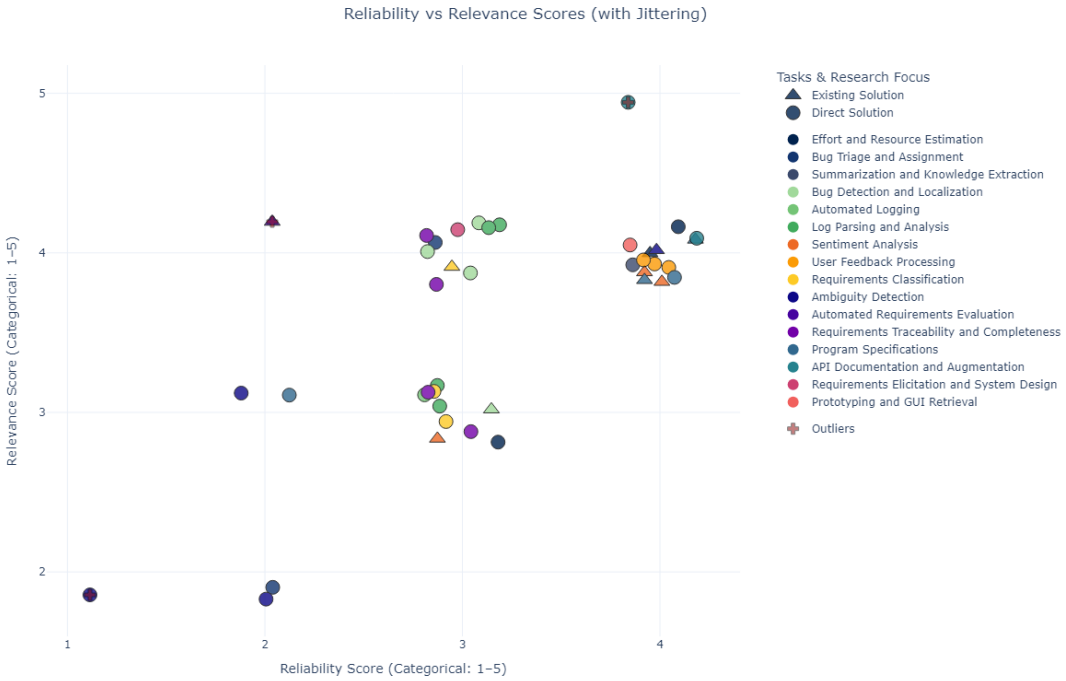



Figure 8: Reliability and Relevance Scores Across Tasks and Research Focus

These overarching patterns provide a foundation for a closer examination of the identified clusters.

### 5.3.3 Cluster Patterns of Reliability and Relevance

The following analysis delves into distinct groups of papers, focusing on those characterized by **moderate reliability and relevance (3,3)**, **moderate reliability and high relevance (3,4)**, and **high reliability and high relevance (4,4)**. Additionally, a smaller group with **low reliability and relevance ($\leq$2,$\leq$3)** and a set of **outlier papers** are briefly discussed. Finally, variations across studies addressing the same tasks but falling into different clusters are explored to uncover task-specific evaluation differences.

**High Reliability and High Relevance (4,4)** This cluster represents evaluations with strong methodological rigor and clear alignment with real-world applicability. Studies in this group ([82, 11, 37, 1, 10, 81, 78, 30, 86, 87, 32, 77, 18, 52]) consistently demonstrate well-validated ground truths, multiple comparative baselines, task-specific metrics, robust validation strategies (e.g., cross-validation, ablation studies, and user studies), and diverse, well-annotated datasets. These strengths contribute to reproducibility and practical relevance, setting a benchmark for evaluation practices.

**Moderate Reliability and High Relevance (3,4)** This cluster includes evaluations that excel in relevance but exhibit moderate reliability due to challenges in specific methodological aspects. Common strengths include manually validated ground truths ([22, 9, 41, 34, 45, 84, 17, 85, 56, 38]), often supported by systematic annotation processes, and diverse comparative baselines tailored to task requirements. Metrics are typically well-aligned with task objectives, ensuring meaningful performance assessment. However, recurring limitations are evident in dataset representativeness, including restricted diversity or small sample sizes, and occasional ambiguity in ground truth annotation processes. Despite these challenges, the evaluations offer valuable practical insights and maintain a strong focus on real-world applicability.

**Moderate Reliability and Moderate Relevance (3,3)** Studies in this cluster achieve balanced but modest performance in both reliability and relevance. While they generally meet baseline methodological standards, recurring weaknesses include reliance on simplistic baselines, limited dataset representativeness, and ambiguities in ground truth validation processes ([48, 15, 14, 40, 80, 19, 43, 42, 92, 3]). Strengths in these evaluations are often seen in the systematic use of human-validated datasets and task-specific evaluation metrics. However, the datasets frequently lack diversity, and the metrics, while task-relevant, are often insufficient to fully capture the nuances of the evaluated tasks. Improvements in dataset design, baseline robustness, and the granularity of evaluation metrics are necessary to enhance the reliability and relevance of these evaluations.

**Low Reliability and Low Relevance ($\leq$2,$\leq$3)** This group highlights studies with significant methodological weaknesses, including poorly defined ground truths, limited or absent baselines, reliance on simplistic or inappropriate metrics, inconsistent validation methods, and small or synthetic datasets ([76, 51, 23, 47]). These evaluations face challenges in both reproducibility and alignment with real-world applicability, necessitating substantial methodological improvements across all aspects.

**Outliers** Three notable outliers deviate from the observed clusters. One study on *API Documentation and Program Specification Generation* ([44]) demonstrates

exceptionally high reliability and relevance. This study excels through the use of **expert-validated specifications** as ground truth, ensuring rigorous correctness and accuracy. It employs **multiple comparative baselines**, including tools like **Houdini**, **Daikon**, and **AutoSpec**, and evaluates results using task-specific metrics such as:

- **Number of Verifier Calls:** Measuring the efficiency of the verification process.
- **Success Probability:** Quantifying the likelihood of successfully generating valid specifications across multiple attempts.
- **User Ratings:** Reflecting human expert evaluation of the semantic clarity and correctness of specifications, assessed through a structured user study involving 15 Ph.D. students with expertise in Java programming and formal verification. The evaluation employed a Likert Scale (1–5) with predefined criteria for clarity, correctness, and completeness, ensuring transparency and consistency in scoring. Ratings were aggregated to minimize bias, providing a robust assessment of the semantic quality of the generated specifications.

The study combines comparative analysis and user surveys, contributing to both methodological rigor and practical applicability. Additionally, the inclusion of diverse datasets (SV-COMP, SpecGenBench, and Defects4J) enhances robustness and generalizability across a variety of real-world scenarios.

Another study on *Automated Requirements Evaluation* ([60]) exhibits strong performance and high relevance (3 for reliability, 4 for relevance), effectively comparing ChatGPT's outputs against human assessments and a baseline tool (AQUSA) using multiple effectiveness metrics. However, the small sample size of only 11 requirements raises concerns about reproducibility and generalizability, and the absence of detailed configuration settings (e.g., temperature values for prompting) further limits transparency.

Conversely, a study on *Ambiguity Detection* ([50]) scores poorly on both dimensions, exhibiting significant methodological shortcomings across multiple aspects. The evaluation was limited to just three examples demonstrating intra-domain ambiguity detection for words. Despite the absence of quantitative ground truth, no effort was made to assess cluster quality, nor was there any qualitative evaluation conducted on a larger sample size. Even for the three examples provided, the evaluation process lacked transparency and was not adequately described.

It is worth noting that the same research group later published an additional paper ([51]), introducing a web tool for their ambiguity detection solution. While this subsequent work provides access and transparency for the validation of the tool, it does not replace the need for a rigorous and systematic evaluation of the original methodology.

These outliers underscore both the **potential and variability** of evaluation practices across tasks and research focuses. High-performing groups excel in **dataset**

**diversity, validation methodologies, and task-specific metrics**, while lower-performing groups consistently struggle with **ground truth clarity, dataset limitations, and validation rigor**.

### 5.3.4 Task-Specific Contrasts in Evaluation Strategies

**Comparison of Requirements Classification Studies** While all three studies (Hey et al. [19] (3,3), Luo et al. [43] (3,3), and El-Hajjami, Fafin, and Salinesi [17] (3,4)) address requirements classification, they exhibit notable differences in their evaluation strategies. Hey et al. [19] relies on the PROMISE dataset, created in 2007, which may not fully capture the characteristics of contemporary software requirements. In contrast, Luo et al. [43] incorporates the NFR-SO dataset, significantly increasing dataset quantity and reflecting more current data trends. However, the StackOverflow data used in NFR-SO lacks alignment with the structured nature of formal software requirements, limiting its applicability representativness for the task beeing adressed.

In comparison, El-Hajjami, Fafin, and Salinesi [17] stands out due to its **cross-dataset validation across five diverse datasets** and its **ability to capture evaluation insights even for ambiguous requirements**, reflecting a nuanced assessment approach better aligned with real-world scenarios. This distinction is particularly relevant as, in real-world settings, requirements often cannot be neatly classified as purely functional or non-functional, underscoring the importance of evaluating performance under such ambiguous conditions. While its small dataset size constrained its reliability score to 3, the diversity of datasets and the robustness of cross-dataset testing suggest it could have reasonably been scored as a 4, highlighting the inherent smoothness of scoring boundaries for individual studies.

**Comparison of Ambiguity Detection Studies** Among the four studies addressing *Ambiguity Detection*, Ezzini et al. [11] (4,4) stands out with its robust evaluation strategy, employing **diverse datasets**, **expert annotations with inter-annotator agreement**, and **recall-focused metrics** to ensure that ambiguous cases are effectively flagged for review. Its use of **cross-validation**, **separate test splits**, and **diverse baseline comparisons** enhances confidence in the evaluation's generalizability and reliability. The study emphasizes the criticality of ambiguity detection in early project stages, where unresolved ambiguities can lead to significant costs in downstream phases. While reliance on manual annotation poses scalability limitations, the evaluation's focus on a large set of industrial requirements significantly strengthens its practical relevance.

In contrast, the other three studies exhibit significant methodological limitations. Moharil and Sharma [50] (1,2) and Moharil and Sharma [51] (2,3) suffer from a lack of **standardized validation metrics**, **absence of baseline comparisons**, and an overreliance on **qualitative analysis**, undermining reproducibility and generalizability. Additionally, the reliance on corpora such as Wikipedia reduces the applicability of their findings to real-world requirements engineering

scenarios. Similarly, Wang et al. [76] (2,2) evaluates its solution on an **undersampled dataset** with an artificial class balance, which diminishes alignment with real-world requirements distributions and limits the evaluation's external validity.

In comparison, Ezzini et al. [11] excels not only in dataset diversity, methodological rigor, and robust baseline comparisons but also in aligning evaluation metrics and validation techniques with the practical needs of requirements engineering. This study offers deeper insights into ambiguity detection's implications for downstream project phases, highlighting its substantial relevance in real-world scenarios.

**Comparison of Program Specification Generation Studies** Among the four studies addressing *Program Specification Generation*, Yang et al. [81], Xie et al. [78], and Endres et al. [10] (all scored 4,4) exhibit robust evaluation methodologies, while Mandal et al. [47] (2,3) demonstrates significant shortcomings in comparison.

The stronger studies share common strengths in their evaluation approaches, including a focus on transparent dataset construction and robust performance benchmarking. For example, Yang et al. [81] emphasizes a well-annotated dataset with high inter-annotator agreement, while Endres et al. [10] leverages expert-verified annotations. These processes enhance confidence in the reliability of their ground truth data.

In terms of baseline comparisons, Yang et al. [81] evaluates performance against diverse baselines, such as **Opiner**, **BERT-family models**, and **CostSensBERT**. Similarly, Xie et al. [78] benchmarks **15 state-of-the-art LLMs** against tools like **Jdoctor** and **DocTer**, and Endres et al. [10] employs a broad range of benchmarks, including **TOGA**, **Daikon**, **GPT-family models**, and **StarChat**. This diversity ensures comprehensive performance assessments across multiple contexts.

The metrics employed by these studies align closely with real-world challenges in program specification. Yang et al. [81] adopts class-balanced metrics such as **Weighted Precision**, **Recall**, **F1 Score**, **MCC**, and **AUC** to address data imbalances effectively. Endres et al. [10] further extends this alignment with task-specific metrics like **Accept@K**, which reflects the likelihood of developers selecting valid postconditions among the top k outputs, and **Bug-Completeness**, which measures a postcondition's ability to discriminate between correct and incorrect implementations using both natural and artificial mutants. These metrics provide nuanced insights into debugging and verification tasks, highlighting robustness and real-world applicability.

While Xie et al. [78] employs general classification metrics such as **Accuracy**, **Precision**, **Recall**, and **F1 Score**, these are complemented by **semantic equivalence validation** and **cross-validation with Few-Shot Learning (FSL)**. Despite this, limited transparency in manual review processes and failure analysis represents a minor weakness in their methodology.

Qualitative analysis also plays a significant role in these studies. For example, Yang et al. [81] includes detailed error analysis, while Xie et al. [78] conducts comparative failure diagnosis, providing deeper interpretative insights into the results. Cross-validation techniques, such as **10-fold validation** [81] and **multi-dataset evaluations** [78, 10], further bolster the reliability of their findings.

In contrast, Mandal et al. [47] suffers from multiple methodological deficiencies. The evaluation relies on a dataset of **250 extracted specifications**, but its creation and the treatment of non-specifications are **poorly documented**. Metrics are limited to basic classifications (**Precision**, **Recall**, **F1 Score**), which fail to address contextual relevance or task-specific needs. Additionally, the absence of diverse baselines, with only **PracExtractor** as a comparator, significantly undermines the interpretability and generalizability of the study's conclusions.

**Comparison of Effort and Resource Estimation Studies** Within *Effort and Resource Estimation*, three reviewed studies share several strengths yet also reveal distinct methodological choices that shape evaluation reliability and relevance. A key strength across all of them is the use of **cross-repository validation** [1, 14, 37], acknowledging the practical challenge of estimating effort in projects lacking extensive historical data. They also evaluate on a **large set of projects**, with two incorporating both open-source and industrial contexts [1, 37] to broaden their applicability across differing work practices and schedules supporting the generalizablity of their findings.

Despite these similarities, the studies diverge in crucial ways in their evaluation strategies. The first paper [1] evaluates the problem as a classification task, which aligns well with the real-world need to not only achieve accurate predictions but also prioritize the rate of "fair enough" estimates—essential for practical decision-making in agile workflows. This evaluation is further strengthened by the use of metrics such as AUC-ROC and F-score, which are particularly effective for handling the imbalanced nature of effort estimates. In contrast, the other two studies adopt regression-based evaluations. Among these, [37] enhances its evaluation by complementing Mean Absolute Error (MAE) with PRED(50), a metric that explicitly assesses the proportion of predictions falling within an acceptable tolerance, thereby addressing the critical real-world requirement of providing practically usable estimates. Fu and Tantithamthavorn [14], however, primarily relies on MAE, which, while effective for measuring absolute accuracy, does not evaluate how well predictions align with the need for "good enough" estimates in agile settings.

They also differ in defining **ground truth**. Two studies [1, 37] use actual effort data gleaned from project records, improving the objectivity and reliability of their evaluations. Fu and Tantithamthavorn [14], however, bases its ground truth on story points, which, while commonly used in agile workflows, reflects the inherent difficulty of effort estimation as a task. This challenge, particularly in the early phases of a project, is prone to errors, as also highlighted in the small-scale user study conducted in [14].

Additionally, [14] is the only study to complement its evaluation with a user study, albeit limited in scale, comparing the perceived supportiveness of effort estimates with and without the inclusion of explainability. This addition offers a unique perspective on how stakeholders perceive the utility of AI-assisted predictions in practical settings, adding qualitative insights that are absent in the other studies.

Overall, Yang et al. [81], Xie et al. [78], and Endres et al. [10] demonstrate well-rounded evaluation methodologies characterized by **diverse datasets**, **robust baseline comparisons**, **task-specific and class-balanced metrics**, and **transparent validation processes**. Their focus on metrics like **Accept@K** and **Bug-Completeness** bridges the gap between experimental benchmarks and real-world software engineering workflows. Conversely, Mandal et al. [47] falls short in addressing these critical aspects, limiting the reliability and applicability of its findings.

# 6 Addressing the Research Questions

## 6.1 Implications for the Reliability of Evaluation Practices

This subsection addresses the first research question (Section 1.2). As defined in Section 4.2.2, reliability measures the consistency, reproducibility, and scientific rigor of evaluation strategies. A highly reliable evaluation produces stable results across datasets, experiments, and environments while adhering to transparent and well-documented methodologies. Applying the reliability scoring system described in Section 4.2.2, the mean score across the reviewed studies is moderate (3.2). This reflects adequate methodological foundations but also highlights significant variability and room for improvement.

*RQ1:* How reliable are the evaluation strategies used in LLM-based software engineering research?

The reliability of evaluation strategies in LLM-based SE research is moderate, with significant variability across tasks. As discussed in Sections 4.2.2 and 5.2.1, recurring limitations in dataset design, ground truth creation, and baseline comparisons often undermine the reproducibility and consistency of evaluations. Addressing these issues is essential for ensuring reliable and scientifically rigorous evaluation practices in future research.

Drawing on the detailed analysis of limitations summarized in Section 5.2.1, this subsection examines the reliability of evaluation strategies across three critical dimensions: datasets, ground truth creation, and baseline comparisons.

### Dataset Design

The reliability of evaluation strategies is highly dependent on the quality and diversity of datasets. Dataset limitations, as detailed in Table 10 in Section 5.2.1, frequently undermine reliability by introducing biases or reducing reproducibility.

- **Narrow Dataset Scope:** Tasks such as *Requirements Traceability* and *Bug Detection* often relied on datasets limited to specific domains, such as open-source projects or Python-based systems [38, 9]. These limitations reduce the generalizability of results and their consistency across different contexts.
- **Insufficient Dataset Volume:** Small dataset sizes, such as the 51 bug reports used in Isotani et al. [23], further restrict the reliability of findings by failing to account for variability across larger or more diverse datasets.

Improving dataset genralizablity and representativeness, as highlighted in Table 11, is crucial for enhancing the reproducibility and robustness of evaluation outcomes.

### Ground Truth Creation

Ground truth creation is a critical component of reliable evaluations. Table 12 in section 5.2.1 highlights recurring challenges in this area, including:

- **Lack of Transparency:** Tasks like *Ambiguity Detection* frequently lacked systematic documentation of ground truth creation processes. For instance, Moharil and Sharma [50] relied on qualitative assessments without quantitative validation, reducing confidence in reproducibility.
- **Annotation Inconsistencies:** In *Log Parsing and Analysis*, Ma et al. [45] did not provide sufficient transparency regarding their annotation process, raising concerns about the reliability of their results.
- **Reliable Practices in High-Performing Tasks:** By contrast, high-performing tasks such as *API Documentation and Augmentation* (Yang et al. [82]) demonstrated reliable practices through transparent annotations and validation strategies, including expert inter-annotator agreement.

Clear and reproducible ground truth processes are essential to ensure consistent evaluation outcomes.

### Baseline Comparisons

Baseline diversity is another critical factor for reliability, as discussed in Section 5.2.1 and summarized in Table 13.

- **Limited Baseline Comparisons:** Studies like Mandal et al. [47] in *Program Specifications* relied on single baselines, which restricted the ability to rigorously compare performance across methods.

– **Absence of Baselines:** In some cases, such as *Ambiguity Detection* (Moharil and Sharma [50]), evaluations lacked baseline comparisons altogether, undermining the reproducibility and robustness of the reported findings.

The inclusion of diverse and meaningful baselines is essential for ensuring reliable evaluations, as it provides a comprehensive understanding of a solution's performance across different conditions.

#### Reliability Across Tasks

The variability in reliability across tasks, as discussed in Section 5.2.1, highlights systemic challenges:

– **High-Performing Tasks (13 Studies) (**$3.6 \leq$ **mean reliability** $\leq 4.0$**):** Tasks such as *API Documentation and Augmentation* demonstrated strong reliability through robust dataset design, transparent ground truth processes, and rigorous baseline comparisons.

– **Moderate-Performing Tasks (21 Studies) (**$3.0 \leq$ **mean reliability** $< 3.6$**):** Tasks such as *Requirements Traceability* adhered to basic methodological standards but showed weaknesses in dataset representativeness and baseline diversity.

– **Low-Performing Tasks (7 Studies) (mean reliability** $< 3.0$**):** Tasks such as *Ambiguity Detection* faced significant methodological weaknesses, including unrepresentative datasets and insufficient validation strategies.

### 6.2 Implications for the Relevance of Evaluation Practices

This subsection addresses the second research question (Section 1.2). Relevance, as defined in Section 4.2.2, measures the alignment of evaluation strategies with practical and real-world challenges in software engineering (SE). Achieving high relevance requires representative datasets, task-specific metrics, and validation strategies that reflect the variability, edge cases, and constraints encountered in real-world scenarios.

*RQ2:* How relevant are the evaluation strategies in reflecting real-world software engineering needs?

Across the reviewed studies, the mean relevance score is 3.59, indicating moderate to high alignment with real-world needs. Many evaluations effectively capture practical challenges; however, key gaps remain. These include reliance on constrained or unrepresentative datasets, the use of generic metrics that fail to capture task-specific needs, and insufficient validation strategies. Addressing these gaps is crucial to delivering actionable insights and improving real-world applicability.

Drawing on the detailed analysis presented in Section 5.2.1, this subsection explores the relevance of evaluation strategies across three critical dimensions: datasets, metrics, and validation strategies.

#### Datasets

Datasets play a foundational role in determining the relevance of evaluations by shaping their alignment with real-world data distributions and practical challenges. As highlighted in Table 10, many studies demonstrated strengths and weaknesses in dataset design:

- **High-Performing Studies:** Tasks such as *API Documentation and Augmentation* (Yang et al. [82]) and *Effort and Resource Estimation* (Li et al. [37]) employed expert-validated datasets and representative real-world examples. These datasets ensured evaluations captured practical scenarios and actionable outcomes.
- **Moderately Performing Studies:** Tasks like *Requirements Classification* used outdated (2007 PROMISE) [19] or dataset not fully aligned with tasks data distribution like StackOverflow [43]
- **Low-Performing Studies:** Tasks such as *Ambiguity Detection* (Moharil and Sharma [50]) non-representative datasets (Wikipedia), reducing their applicability to real-world settings. For *Duplicate Bug detection* [23] relied on a too small test set of only 51 Bug reports with biased and unrestnative class distribution (duplicates/nonduplicates).

To improve relevance, future evaluations should prioritize datasets that reflect real-world representativeness of the test and validation data.

#### Metrics

Metrics are essential for quantifying the performance of LLM-based solutions in ways that align with task-specific objectives and real-world priorities. The analysis in Table 14 highlights both effective practices and common limitations:

- **Task-Specific Metrics:** High-relevance studies, such as *Effort and Resource Estimation* (Li et al. [37]), incorporated complementary metrics like tolerance ranges to evaluate estimation accuracy, directly addressing practical decision-making needs.
- **Misaligned Metrics:** In tasks like *Bug Triage and Assignment* (Isotani et al. [23]), reliance on mean average precision (MAP) without cutoffs limited the practical relevance of ranking insights.
- **Underutilized Metrics:** In *Ambiguity Detection* (Moharil and Sharma [50]), evaluations lacked cluster quality metrics, reducing their ability to address key aspects of ambiguity in requirements engineering.

Future evaluations should ensure that metrics are closely aligned with task-specific goals, particularly in addressing practical considerations like scalability, usability, and interpretability.

### Validation Strategies

Validation strategies ensure that evaluations adequately capture real-world challenges by testing solutions across diverse conditions and scenarios. As summarized in Table 15, studies displayed varying levels of alignment with real-world needs:

- **High-Performing Strategies:** Studies in *Log Parsing and Analysis* (Liu et al. [40]) conducted ablation studies and feature impact analyses to evaluate model robustness and relevance.
- **Limited Strategies:** In tasks like *Coreference Detection* (Wang et al. [76]), evaluations relied on artificially balanced datasets, which failed to capture the complexity and variability of real-world data distributions.
- **Qualitative Approaches:** Studies such as Xu et al. [80] performed qualitative error analysis to identify critical performance gaps, demonstrating the value of structured qualitative methods in improving practical insights.

Broader adoption of comprehensive validation methods, including real-world simulations and user studies, would significantly enhance the relevance of evaluations.

### Relevance Across Tasks

The relevance of evaluation strategies varied across tasks, as evidenced by their mean relevance scores:

- **High-Performing Tasks (**$3.66 \leq$ **mean relevance** $\leq 4.5$**):** Tasks such as *API Documentation and Augmentation*, *Prototyping and GUI Retrieval*, and *User Feedback Processing* demonstrated strong alignment with real-world challenges, supported by representative datasets and task-specific metrics.
- **Moderately Performing Tasks (**$3.0 \leq$ **mean relevance** $< 3.66$**):** Tasks such as *Bug Detection* and *Program Specifications* exhibited strengths in component-level evaluations but faced limitations in dataset diversity or metric alignment.
- **Low-Performing Tasks (mean relevance** $< 3.0$**):** Tasks such as *Ambiguity Detection* and *Coreference Detection* relied on overly simplistic datasets and metrics, reducing their applicability to real-world scenarios.

## 6.3 Addressing Key Gaps and Enhancing Evaluation Practices

Building on the findings discussed in Sections 6.1 and 6.2, this analysis identifies major shortcomings in current practices and explores how future evaluations can achieve greater rigor and practical applicability.

*RQ3:* What are the key gaps and limitations in current evaluation strategies and how can they be addressed?

**Key Gaps:**

- **Dataset Limitations:** Narrow, outdated, and poorly documented datasets reduce generalizability, transparency, and representativeness, limiting reproducibility and alignment with real-world scenarios.
- **Underdeveloped Task-Specific Best-Practices:** Variations in evaluation strategies reveal potential for combining strengths, but inconsistent practices hinder reproducibility and comparability.
- **Neglect of Qualitative Methods:** The lack of qualitative evaluations limits insights into solution usability and real-world impact.
- **NSE vs. PES Approaches:** PES studies tend to exhibit stronger reliability due to standardized datasets and comprehensive baselines, while NSE studies innovate but often face challenges with small or task-specific datasets and unclear annotations. There is a missed opportunity for PES studies to integrate novel solutions proposed in NSE studies into their baselines and for NSE studies to adopt PES methodological rigor.
- **Overlooking LLM-Specific Traits:** Most evaluations treat LLMs as agnostic tools, neglecting variability, non-determinism, and domain-specific factors that affect reliability and relevance.

**Recommendations:** Improve dataset practices, develop task-specific best practices, and incorporate qualitative feedback. Leverage strengths across NSE and PES approaches and integrate LLM-specific traits into evaluation frameworks to enhance rigor and real-world alignment.

### Gaps in Current Evaluation Practices

#### Dataset-Related Limitations

Dataset limitations emerged as the most pervasive issue affecting both reliability and relevance across multiple tasks and research focus categories. While such limitations were present in both NSE and PES studies, they were notably more pronounced in the NSE category, where novel solutions often depend on newly

created datasets. As summarized in Table 11, three major themes were observed: generalizability, transparency, and representativeness. These limitations directly impact the consistency and applicability of evaluation outcomes.

**Generalizability** Generalizability issues predominantly affect the reliability of evaluation practices by limiting the extent to which findings can be reproduced or extended to new scenarios. For example, evaluations in *Bug Detection* and *Requirements Traceability* often relied on datasets constrained to open-source repositories [9, 38, 92]. Similarly, Liu et al. [41] employed a limited sample of widgets for testing, and Huang et al. [22] used a narrowly scoped crash report dataset. These choices reduce the diversity of test cases and hinder evaluations from capturing the variability encountered in real-world settings.

**Transparency** A lack of transparency in dataset creation processes further undermines the reliability of evaluations by impeding reproducibility. For example, Mandal et al. [47] did not clearly document how the non-specification class was constructed for their test set, raising concerns about the validity of their comparisons. Similar issues were observed in *Ambiguity Detection* and *Requirements Elicitation*, where the origins of datasets or validation processes were either unclear or absent [85, 51]. Without clear documentation, independent reproduction and validation of these results become challenging.

**Representativeness** Issues with dataset representativeness primarily impact the relevance of evaluation practices by limiting their alignment with real-world contexts. For example, studies in *Program Specification* and *Requirements Classification* often relied on small or outdated datasets, such as the 2007 PROMISE dataset used by Hey et al. [19]. In *Ambiguity Detection*, reliance on the Wikipedia Corpus [50, 51] failed to account for the unique linguistic characteristics of software requirements. Similarly, Luitel, Hassani, and Sabetzadeh [42] simulated incompleteness in requirements datasets, which may not accurately reflect real-world gaps. These limitations reduce the practical applicability of evaluations and risk overestimating the effectiveness of solutions in diverse, real-world scenarios.

It is important to note that efforts are being made within the research community to address dataset limitations, as observed in the development of new resources to support evaluations. As discussed in Section 4.2.1, four papers ([65, 25, 70, 13]) propose new datasets as part of their contributions, alongside either the development or evaluation of solutions. Moreover, the research focus category "Creation of SE Evaluation Resources" encompasses 29 papers that provide various types of evaluation resources, including but not limited to datasets. These works collectively indicate an awareness of the importance of high-quality datasets, benchmarks, and related resources for enhancing the reliability and relevance of evaluation practices.

### Developing Task-Specific Best Practices for Evaluation

Task-specific best practices for evaluation are essential for achieving methodological rigor and ensuring insights align with the objectives of real-world software engineering tasks. By systematically combining strengths from diverse evaluation strategies, best practices not only enhance reliability and relevance but also improve the comparability of results across studies. This comparability is vital for cumulative knowledge building and the broader adoption of effective solutions.

The effort estimation task provides a concrete example of how such best practices can be developed. Drawing from Section 5.3.4, the reviewed studies highlight complementary strengths in addressing generalizability, evaluation metrics, and the alignment of ground truth with task objectives. By synthesizing these strengths, a robust framework for evaluating effort estimation solutions can be constructed, offering a blueprint for developing task-specific best practices across other software engineering domains.

The three reviewed studies share complementary strengths that inform best practices. A unifying feature is their use of cross-repository validation, a crucial step for assessing generalizability in high-uncertainty situations where limited project information is available. Incorporating both open-source and industrial datasets enhances the applicability of evaluations by addressing variability across different work environments.

To develop a comprehensive best practice evaluation for effort estimation, the following components emerge:

1. **Cross-Repository Validation:** Evaluation strategies should assess generalizability in scenarios with limited historical data, a challenge common in early project stages. Cross-repository testing, as demonstrated by the studies, ensures robustness in such high-uncertainty settings. Combining open-source and industrial datasets further supports evaluations across diverse real-world conditions.
2. **Balanced Metrics:** Best practices should include statistical measures such as Mean Absolute Error (MAE) for accuracy, alongside practical usability metrics like PRED(50) and AUC-ROC. These metrics not only evaluate precision but also assess the acceptability of estimates within reasonable tolerances, directly addressing the decision-making needs of practitioners.
3. **Ground Truth Alignment with Objectives:** Using objective measures of actual effort, as seen in Alhamed and Storer [1] and Li et al. [37], better aligns with the goal of supporting decision-makers with estimates backed by reliable, real-world data. This approach avoids the variability and subjectivity inherent in story point-based ground truths.
4. **Assessment of Perceived Supportiveness:** Evaluations should include qualitative assessments of how practitioners perceive the support provided by solutions, particularly in high-uncertainty contexts. For instance, the user

study in Fu and Tantithamthavorn [14] demonstrates how feedback on perceived utility and ease of adoption offers valuable insights beyond statistical metrics.

By combining these elements, task-specific best practices create a foundation for evaluations that not only ensure methodological rigor but also yield insights directly relevant to the task's real-world objectives. Moreover, standardizing such practices fosters comparability across studies, enabling cumulative knowledge building and more effective application of solutions.

**Broader Implications Across Tasks.** While effort estimation illustrates the potential for developing best practices, this principle extends to other domains as well. As shown in Section 5.3.4, tasks such as *Program Specification Generation* and *Ambiguity Detection* also benefit from tailored evaluation strategies. For example, metrics like Accept@K and Bug-Completeness in specification generation capture nuanced challenges that generic metrics overlook. Similarly, recall-focused metrics and expert-annotated datasets strengthen evaluations in ambiguity detection, ensuring alignment with the task's practical objectives.

By promoting consistent evaluation practices within tasks, the research community can enable more reliable cross-study comparisons. This comparability is essential for synthesizing insights that advance task-specific understanding, support benchmarking, and ultimately improve the applicability of solutions to real-world software engineering challenges.

#### Integrating Qualitative Insights into Evaluation

A further observation from the reviewed corpus is the limited use of qualitative methods—such as user studies or qualitative analysis of results. While most studies focus primarily on numerical measures (e.g., precision, recall, F1, MAE), only five were found to incorporate user studies or qualitative review of prediction results and errors to gain practical feedback on solution utility and ease of adoption. Table 16 and the validation issues in Section 5.2.1 highlight that these user studies were typically of small to medium size, limiting the depth and representativeness of the qualitative insights.

For instance, Fu and Tantithamthavorn [14] included a small-scale user study to gauge how agile practitioners perceived the tool's "supportiveness" in effort estimation, providing practical feedback that went beyond its primary metric of Mean Absolute Error (MAE). Meanwhile, other studies with robust quantitative evaluations—such as Ma et al. [44], Kou, Chen, and Zhang [32], Wang et al. [77], and Kolthoff, Bartelt, and Ponzetto [30]—also integrated user studies or qualitative analyses of subsets of their results, reinforcing their findings with real-world usability insights. Although these user studies were often constrained by sample size or lacked standardization, they offered valuable glimpses into how participants interacted with the proposed solutions in realistic settings.

**Recommendation: Complement Quantitative Metrics with Qualitative Methods.** Future evaluation strategies should pair solid quantitative assessments with carefully designed qualitative components, such as user surveys and qualitative error analysis. For example, collecting feedback on perceived usability, clarity of system outputs, or ease of integration into existing workflows can illuminate important usability or adoption issues that purely numerical metrics may overlook. By systematically embedding these user-facing perspectives, researchers can produce evaluations that capture both technical performance and practical viability, ultimately aligning more closely with the needs of software engineering practitioners.

### Contrasting Evaluation Approaches in NSE and PES

As outlined in Section 5.2.2, the two primary research focuses—*Development of New SE Solutions (NSE)* and *Performance Assessment of Existing Solutions (PES)*—show different strengths and weaknesses. While both categories exhibit diverse validation and metrics practices, studies from the PES category tend to deliver *more consistent reliability* due to controlled and statistically grounded validation, clearer documentation, and reliance on well-defined datasets. By contrast, NSE studies, despite demonstrating notable innovations (e.g., user studies, comparative analyses), often face additional obstacles such as uncertain annotations, smaller or task-specific datasets, and limited baseline diversity (see Table 18 and Table 17).

These patterns reveal a missed opportunity: PES evaluations should broaden their baselines by incorporating newly proposed solutions from the NSE category, thereby ensuring more comprehensive benchmarking. At the same time, NSE evaluations should leverage the methodical and comparative rigor commonly observed in PES studies. Although this thesis does not specifically examine the extend how PES papers already incorporate NSE solutions as baselines, deeper collaboration between these research focuses stands to benefit both solution development and evaluation methodologies.

### Overlooking LLM-Specific Characteristics in Evaluation Practices

The qualitative review of the selected corpus, as described in Section 4.3.1, revealed that the majority of evaluation strategies treat large language models (LLMs) as agnostic tools. These evaluations primarily focus on task-level performance metrics such as precision, recall, and F1 scores, while often neglecting critical LLM-specific characteristics that could influence evaluation outcomes, including prompt sensitivity, non-determinism, and variability across runs.

This observation emerged from noticing the absence of considerations for LLM-specific characteristics in evaluation practices. However, it is important to note that these observations were not systematically tracked in the review notes. This limitation highlights the need for a more detailed and systematic investigation

into the role of LLM-specific characteristics in evaluation strategies in future research.

**Concrete Observations:**

- **Repetition to Mitigate Variability:** Ronanki, Cabrero-Daniel, and Berger [60] addressed the inherent non-determinism of ChatGPT by repeating evaluations three times and adopting a "best-of-three" approach to determine consistent results. While this method acknowledges variability, it does not systematically analyze the impact of different random seeds or decoding parameters, such as temperature settings.
- **Incorporation of Probability Distributions:** Luitel, Hassani, and Sabetzadeh [42] incorporated BERT's probability distributions into their evaluation methodology for masked language modeling tasks. By varying the number of predictions per masked token, they balanced recall and precision, providing deeper insights into the model's contextual reasoning. However, similar efforts to leverage probability distributions were not observed in other reviewed studies, indicating an underutilization of this approach in evaluation practices.
- **Transparency in Temperature Settings:** Liu et al. [40] demonstrated transparency by explicitly reporting their use of temperature settings (e.g., temperature=0.5) in experiments. They reduced the temperature to 0.4 when outputs failed to meet format expectations. While this transparency aids reproducibility, the study did not systematically explore how variations in temperature or other decoding parameters affect evaluation metrics.

These examples illustrate that although some studies make initial efforts to account for LLM-specific characteristics, these efforts are limited and not systematically integrated into evaluation frameworks.

**Leveraging Insights from LLM Characteristic Exploration** The separate research focus category, "Exploration of LLM Characteristics in SE" discribed in Section 4.2.1 provides valuable insights that could inform and enhance evaluation strategies. For example:

- **Non-Determinism in LLM Outputs:** Ouyang et al. [54] demonstrated that even when temperature is set to 0, LLM outputs like those from ChatGPT can still exhibit variability across runs. This challenges the common assumption that setting temperature to 0 ensures deterministic behavior and highlights the importance of accounting for such variability in evaluation practices.
- **Domain-Specific Pretraining and Vocabulary:** Von der Mosel, Trautsch, and Herbold [73] explored how domain-specific pretraining impacts LLM performance, particularly in tasks like requirements engineering. They examined vocabulary differences and the effectiveness of masked language

modeling in SE contexts, emphasizing the need for evaluation datasets that reflect domain-specific language and challenges.

These studies suggest that incorporating an understanding of LLM-specific behaviors—such as variability and domain adaptation—can lead to more robust and relevant evaluation practices.

#### Recommendations for Advancing Evaluation Strategies

To address the identified gaps in current evaluation practices, the following recommendations are proposed:

1. **Account for Non-Determinism:** Incorporate repeated trials using varied random seeds and decoding parameters. Report variability metrics such as confidence intervals and standard deviations to enhance the reliability and transparency of evaluations. For example, adopting practices from Ouyang et al. [54] can help quantify the extent of variability and ensure that evaluation results are robust across different runs.
2. **Enhance Transparency of Experimental Settings:** Clearly document and report key LLM parameters, including temperature settings and any adjustments made during experiments. Transparency in these settings, as demonstrated by Liu et al. [40], is crucial for reproducibility and allows for a better understanding of how these parameters influence evaluation outcomes.

Future strategies that respect LLM-specific characteristics need to be explored. Insights from papers that explore these characteristics for SE tasks, such as those by Von der Mosel, Trautsch, and Herbold [73] and Ouyang et al. [54], could guide the development of more nuanced and effective evaluation methodologies.

## 7 Conclusion

This thesis investigated the evaluation of Large Language Models (LLMs) in Software Engineering (SE) research, focusing on the reliability and relevance of evaluation strategies. A six-phase systematic approach was employed to select and analyze studies. The review process refined an initial corpus of 396 papers ([20]) to 41 papers focused on non-code-centric SE tasks, ensuring inclusion criteria prioritized task-specific solution evaluations over explorations of LLM characteristics or resource creation. Papers were categorized by task objectives to ensure a coherent review.

Phase 4 involved a systematic qualitative review of key evaluation aspects such as datasets, baselines, metrics, and validation methods. Phases 5 and 6 analyzed the strengths and limitations of these strategies, revealing recurring challenges like dataset representativeness, baseline diversity, and validation consistency.

Phase 6 synthesized these findings to assess overall reliability and relevance, uncover systemic trends, and identify task-specific contrasts. These efforts provided nuanced insights into the impact of evaluation practices and informed answers to the research questions (Sections 6.1, 6.2, and 6.3).

## 7.1 Key Conclusions

The research revealed significant variations in the methodological rigor and real-world applicability of evaluation strategies in LLM-based SE research. A key issue affecting both reliability and relevance is the widespread reliance on readily available, yet narrow and sometimes outdated datasets, such as Stack Overflow, Wikipedia, and open-source repositories. While these datasets offer convenience, they frequently lack representativeness and fail to generalize beyond open-source contexts, limiting their applicability to real-world SE tasks.

Despite these limitations, researchers demonstrated moderate reliability and relevance in their evaluation practices. Many studies adhered to baseline methodological standards, such as aligning datasets with task objectives and creating ground truth data. However, these efforts often involved trade-offs between efficiency and thoroughness, leading to shortcomings such as incomplete documentation of ground truth creation processes and insufficient consideration of dataset generalizability. Reliance on existing datasets frequently constrained these evaluations, prioritizing speed over depth.

Relevance was further affected by the limited alignment of datasets and metrics with the specific characteristics of the tasks being evaluated. For instance, while datasets like Stack Overflow data are somewhat related to tasks such as Requirements Classification, they do not fully match the structured requirements typically encountered in real-world settings, leading to a mismatch in distribution. Metrics, such as Mean Average Precision (MAP), though generally useful, often failed to capture practical priorities—such as ranking top recommendations in tasks like traceability link prediction—highlighting a broader trend across evaluations rather than being an isolated case. Additionally, qualitative methods, such as user studies or manual validation, were rarely employed, limiting the ability to derive contextual and practical insights.

### Implications for the Field

The rapid evolution of LLM-based SE research has driven researchers to make trade-offs between fast-paced solution development and rigorous evaluation practices. While this approach allows researchers to stay aligned with the fast-changing landscape of LLM advancements, it has also led to limitations in evaluation methodologies. These trade-offs often manifest in the reliance on readily available datasets and generic evaluation metrics, which, while expedient, undermine the reliability, relevance, and trustworthiness of evaluation results.

To ensure that LLM-based SE solutions deliver tangible value and sustain long-term progress, it is imperative to establish reliable and trustworthy evalua-

tion practices. Without such practices, evaluations risk falling short of providing meaningful insights, potentially leading to disillusionment in the field and stalling its momentum.

## 7.2 Opportunities for Improvement

Strengthening dataset diversity, fostering collaboration, and embracing task-specific evaluation frameworks present critical opportunities to address these issues.

1. **Addressing Dataset Limitations:** Many SE tasks require data that are not publicly available due to their sensitivity, particularly in industrial contexts. Strategies must be developed to enable the sharing of sensitive company data without compromising confidentiality. Establishing broader collaborations between industry and academia could facilitate the creation of datasets that capture real-world variability and edge cases, enhancing the generalizability of evaluation results.
2. **Developing Task-Specific Best Practices:** The review revealed that within the same task, different studies exhibited unique strengths in their evaluation practices. This diversity creates an opportunity for researchers to synthesize these strengths into task-specific best practices. By fostering closer collaboration between researchers working on similar tasks, the field can develop tailored evaluation frameworks that ensure both reliability and relevance.
3. **Integrating Research from NSE and PES Categories:** Studies in the Performance Assessment of Existing Solutions (PES) category generally exhibited stronger reliability due to their controlled validation processes and robust methodologies. Conversely, Development of New SE Solutions (NSE) studies, while innovative, often lacked comprehensive evaluation frameworks. Incorporating solutions from NSE studies into the baselines of PES studies would provide broader, more empirical assessments and enable further reflection on the effectiveness, scalability, and limitations of emerging approaches. This integration requires PES studies to more broadly explore the field for novel solutions and invest additional effort in reproducing and validating these innovations. Such efforts will enhance the alignment of innovative solutions with real-world requirements and foster more reliable evaluation practices across SE tasks.
4. **Leveraging LLM-Specific Characteristics:** Opportunities are being missed to incorporate insights from research exploring LLM-specific characteristics. For instance, the non-deterministic behavior of LLMs, such as variations in output even when temperature settings are fixed, is rarely accounted for systematically. Incorporating such insights into evaluation methodologies will enhance robustness and better reflect the nuances of LLM behavior.

### Why This Matters

If the field fails to address these gaps and seize these opportunities, it risks undermining trust in LLM-based SE solutions. Poorly grounded evaluations may perpetuate hype without delivering meaningful advancements, ultimately leading to disillusionment. By addressing these challenges and prioritizing rigorous, task-specific, and collaborative evaluation practices, the field can ensure that its findings remain relevant and impactful, supporting long-term progress in both academic and industrial settings.

## 7.3 Recommendations for Trustworthy Evaluations and Where More Caution is Required

The evaluation practices in LLM-based SE research vary in reliability and relevance. Below is a summary of tasks with trustworthy evaluations and guidance for interpreting other results cautiously.

**Tasks with Trustworthy Evaluation Practices** The following tasks exhibit strong evaluation methodologies, including robust datasets, metrics designed to reflect real-world applicability, and rigorous validation practices such as crowdsourced annotations or user studies:

- **API Documentation Augmentation:** Studies such as Yang et al. [81] and Yang et al. [82] employ transparently annotated datasets with high inter-annotator agreement and metrics addressing class imbalance (e.g., Weighted F1, MCC). These evaluations also integrate human validation to provide practical insights, aligning with real-world documentation needs.
- **Sentiment Analysis:** Research by Biswas et al. [3], Zhang et al. [86], and Zhang et al. [87] leverages diverse datasets, macro-F1 and micro-F1 metrics, and detailed error analyses to address challenges like implicit sentiment and subjective annotations. These studies balance methodological rigor with real-world relevance.
- **User Feedback Processing:** Examples such as Wang et al. [77], He et al. [18], and Motger et al. [52] incorporate large-scale, diverse datasets validated by inter-annotator agreement. Tailored metrics like Precision@k and Recall@k assess specific needs (e.g., tag recommendations or feature extraction). They include user studies or qualitative evaluations to align findings with practical applications.
- **Knowledge Summarization:** Kou, Chen, and Zhang [32] evaluates automated summarization for Stack Overflow posts, employing cross-validation, precision, recall, and F1-score for quantitative robustness. A user study provides real-world insights into summarization quality, though its small size limits generalizability. The evaluation addresses diverse post types (e.g., how-to, bug-fixing), enhancing relevance for developer scenarios.

– **GUI Retrieval and Prototyping:** Kolthoff, Bartelt, and Ponzetto [30] integrates rigorous methods, including a high-quality gold standard dataset annotation by multiple independent annotators and validated with inter-annotator agreement. The study incorporates a user study with novice developers, comparing the proposed tool to a baseline under real-world prototyping scenarios. Metrics such as NDCG@k and HITS@k assess ranking quality, while user effort metrics (e.g., number of GUI components added and diversity) evaluate practical efficiency.

These tasks provide evaluation results that decision-makers can trust for further development and application.

**Tasks Requiring Caution** In some cases, evaluations warrant more careful interpretation, especially for tasks where:

– **Context-Specific Workflows:** Tasks often depend on unique company contexts or workflows, which are not captured by widely available datasets (e.g., Stack Overflow, Wikipedia). While these datasets are commonly used for evaluation, they may not reflect the specific industrial environments in which these tasks are applied. Decision-makers should critically assess whether the data used for testing and validation aligns with their own operational contexts and data characteristics.
– **Unclear Ground Truth:** Tasks with subjective or hard-to-define ground truths, such as Ambiguity Detection, are inherently challenging to evaluate. Results for these tasks may be less reliable, requiring careful scrutiny to ensure their applicability to real-world scenarios.

### 7.4 Further Work

This thesis focused on the evaluation strategies of LLM-based solutions for non-code-centric SE tasks, providing critical insights into their reliability and relevance. Building on these findings, the following areas require further investigation:

– **Expanding to Code-Centric Tasks:** Code-centric tasks, including code generation, completion, and repair, were excluded from detailed analysis due to their distinct challenges and evaluation requirements, which warrant dedicated investigation. These tasks represent a substantial portion of LLM applications in SE and involve unique considerations such as execution-based metrics, runtime performance, and functional correctness. Addressing these distinct challenges is necessary to develop a comprehensive understanding of evaluation practices across the full spectrum of SE applications.
– **Exploration of LLM Characteristics in SE:** A distinct category of 56 papers investigates LLM properties, such as robustness, generalizability, and interpretability, within SE contexts. These studies provide essential insights into

how LLM characteristics impact SE tasks and inform the design of more effective evaluation strategies. Reviewing these works will contribute to the development of evaluation methodologies that account for the specific behaviors and nuances of LLMs in SE applications.

– **Assessing SE Evaluation Resources:** Another important area involves the 29 papers that focus on developing datasets, benchmarks, and other evaluation resources for SE tasks. These resources are likely to be widely adopted and will shape future evaluations. Reviewing and assessing the quality and appropriateness of these resources is crucial to ensure that they support robust, reliable, and task-relevant evaluations. Combining insights from these works with research on LLM characteristics will facilitate the creation of standardized and task-specific evaluation frameworks, ultimately improving the reliability and relevance of future evaluations.

By expanding the scope to include code-centric tasks, LLM characteristics, and evaluation resource assessment, future research can address broader challenges in LLM evaluation for SE tasks. These efforts are necessary to advance the field and ensure that LLM-based solutions in Software Engineering deliver reliable, relevant, and impactful outcomes.

### 7.5 Final Reflections

In conclusion, this thesis highlights the critical role of robust evaluation strategies in advancing the reliability and relevance of LLM-based solutions in Software Engineering. By systematically analyzing existing evaluation practices, the study uncovered significant challenges, such as dataset limitations, methodological trade-offs, and mismatches between evaluation metrics and real-world tasks. Addressing these gaps is essential not only to ensure that LLM-based tools meet the practical needs of the SE community but also to sustain trust and momentum in this rapidly evolving field.

The findings emphasize that reliable and task-specific evaluation frameworks are paramount for translating the potential of LLMs into tangible advancements. By fostering collaboration between academia and industry, embracing diverse and representative datasets, and integrating insights into LLM-specific behaviors, the field can overcome existing barriers and unlock new opportunities for impactful research. Looking ahead, this work underscores the importance of prioritizing rigor and relevance in evaluation practices to ensure that LLM-based solutions deliver meaningful and lasting contributions to Software Engineering.

## References

[1] Mohammed Alhamed and Tim Storer. “Evaluation of context-aware language models and experts for effort estimation of software maintenance issues”. In: *2022 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE. 2022, pp. 129–138.

[2] Friedrich L Bauer. "Software Engineering—wie es begann". In: *Historische Notizen zur Informatik* (2009), pp. 72–75.

[3] Eeshita Biswas et al. "Achieving reliable sentiment analysis in the software engineering domain using bert". In: *2020 IEEE International conference on software maintenance and evolution (ICSME)*. IEEE. 2020, pp. 162–173.

[4] Nghi DQ Bui et al. *CodeTF: One-stop Transformer Library for State-of-the-art Code LLM.(May 2023)*. 2023.

[5] Yupeng Chang et al. "A survey on evaluation of large language models". In: *ACM Transactions on Intelligent Systems and Technology* (2023).

[6] Fuxiang Chen et al. "On the transferability of pre-trained language models for low-resource programming languages". In: *Proceedings of the 30th IEEE/ACM International Conference on Program Comprehension*. 2022, pp. 401–412.

[7] Mark Chen et al. "Evaluating large language models trained on code". In: *arXiv preprint arXiv:2107.03374* (2021).

[8] Yizheng Chen et al. "Diversevul: A new vulnerable source code dataset for deep learning based vulnerability detection". In: *Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses*. 2023, pp. 654–668.

[9] Agnieszka Ciborowska and Kostadin Damevski. "Fast changeset-based bug localization with BERT". In: *Proceedings of the 44th International Conference on Software Engineering*. 2022, pp. 946–957.

[10] Madeline Endres et al. "Formalizing natural language intent into program specifications via large language models". In: *arXiv preprint arXiv:2310.01831* (2023).

[11] Saad Ezzini et al. "Automated handling of anaphoric ambiguity in requirements: a multi-solution study". In: *Proceedings of the 44th International Conference on Software Engineering*. 2022, pp. 187–199.

[12] Angela Fan et al. "Large language models for software engineering: Survey and open problems". In: *arXiv preprint arXiv:2310.03533* (2023).

[13] Zhiyu Fan et al. "Automated repair of programs from large language models". In: *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE. 2023, pp. 1469–1481.

[14] Michael Fu and Chakkrit Tantithamthavorn. "GPT2SP: A transformer-based agile story point estimation approach". In: *IEEE Transactions on Software Engineering* 49.2 (2022), pp. 611–625.

[15] Luiz Gomes, Ricardo da Silva Torres, and Mario Lúcio Côrtes. "BERT-and TF-IDF-based feature extraction for long-lived bug prediction in FLOSS: a comparative study". In: *Information and Software Technology* 160 (2023), p. 107217.

[16] Zishan Guo et al. "Evaluating large language models: A comprehensive survey". In: *arXiv preprint arXiv:2310.19736* (2023).

[17] Abdelkarim El-Hajjami, Nicolas Fafin, and Camille Salinesi. “Which AI Technique Is Better to Classify Requirements? An Experiment with SVM, LSTM, and ChatGPT”. In: *arXiv preprint arXiv:2311.11547* (2023).

[18] Junda He et al. “PTM4Tag: sharpening tag recommendation of stack overflow posts with pre-trained models”. In: *Proceedings of the 30th IEEE/ACM International Conference on Program Comprehension*. 2022, pp. 1–11.

[19] Tobias Hey et al. “Norbert: Transfer learning for requirements classification”. In: *2020 IEEE 28th International Requirements Engineering Conference (RE)*. IEEE. 2020, pp. 169–179.

[20] Xinyi Hou et al. “Large language models for software engineering: A systematic literature review”. In: *arXiv preprint arXiv:2308.10620* (2023).

[21] Jie Hu, Qian Zhang, and Heng Yin. “Augmenting greybox fuzzing with generative ai”. In: *arXiv preprint arXiv:2306.06782* (2023).

[22] Yuchao Huang et al. “Crashtranslator: Automatically reproducing mobile application crashes directly from stack trace”. In: *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering*. 2024, pp. 1–13.

[23] Haruna Isotani et al. “Duplicate bug report detection by using sentence embedding and fine-tuning”. In: *2021 IEEE international conference on software maintenance and evolution (ICSME)*. IEEE. 2021, pp. 535–544.

[24] Prithwish Jana et al. “Attention, compilation, and solver-based symbolic analysis are all you need”. In: *arXiv preprint arXiv:2306.06755* (2023).

[25] Matthew Jin et al. “Inferfix: End-to-end program repair with llms”. In: *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2023, pp. 1646–1656.

[26] Jai Kannan. “Can LLMs Configure Software Tools”. In: *arXiv preprint arXiv:2312.06121* (2023).

[27] Staffs Keele et al. *Guidelines for performing systematic literature reviews in software engineering*. 2007.

[28] Barbara Kitchenham, Lech Madeyski, and David Budgen. “SEGRESS: Software engineering guidelines for reporting secondary studies”. In: *IEEE Transactions on Software Engineering* 49.3 (2022), pp. 1273–1298.

[29] Takashi Koide et al. “Detecting phishing sites using chatgpt”. In: *arXiv preprint arXiv:2306.05816* (2023).

[30] Kristian Kolthoff, Christian Bartelt, and Simone Paolo Ponzetto. “Data-driven prototyping via natural-language-based GUI retrieval”. In: *Automated software engineering* 30.1 (2023), p. 13.

[31] Stefan Kombrink et al. “Recurrent Neural Network Based Language Modeling in Meeting Recognition.” In: *Interspeech*. Vol. 11. 2011, pp. 2877–2880.

[32] Bonan Kou, Muhao Chen, and Tianyi Zhang. “Automated summarization of stack overflow posts”. In: *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE. 2023, pp. 1853–1865.

[33] Teven Le Scao et al. “Bloom: A 176b-parameter open-access multilingual language model”. In: (2023).

[34] Jaehyung Lee, Kisun Han, and Hwanjo Yu. “A light bug triage framework for applying large pre-trained language model”. In: *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*. 2022, pp. 1–11.

[35] Caroline Lemieux et al. “Codamosa: Escaping coverage plateaus in test generation with pre-trained large language models”. In: *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE. 2023, pp. 919–931.

[36] Jingxuan Li et al. “Toward less hidden cost of code completion with acceptance and ranking models”. In: *2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE. 2021, pp. 195–205.

[37] Yue Li et al. “Fine-SE: Integrating Semantic Features and Expert Features for Software Effort Estimation”. In: *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering*. 2024, pp. 1–12.

[38] Jinfeng Lin et al. “Traceability transformed: Generating more accurate links with pre-trained bert models”. In: *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE. 2021, pp. 324–335.

[39] Chao Liu et al. “Improving chatgpt prompt for code generation”. In: *arXiv preprint arXiv:2305.08360* (2023).

[40] Yilun Liu et al. “Interpretable online log analysis using large language models with prompt strategies”. In: *Proceedings of the 32nd IEEE/ACM International Conference on Program Comprehension*. 2024, pp. 35–46.

[41] Zhe 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*. 2024, pp. 1–12.

[42] Dipeeka Luitel, Shabnam Hassani, and Mehrdad Sabetzadeh. “Improving requirements completeness: Automated assistance through large language models”. In: *Requirements Engineering* 29.1 (2024), pp. 73–95.

[43] Xianchang Luo et al. “PRCBERT: Prompt Learning for Requirement Classification using BERT-based Pretrained Language Models”. In: *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*. 2022, pp. 1–13.

[44] Lezhi Ma et al. “SpecGen: Automated Generation of Formal Program Specifications via Large Language Models”. In: *arXiv preprint arXiv:2401.08807* (2024).

[45] Lipeng Ma et al. “Knowlog: Knowledge enhanced pre-trained language model for log understanding”. In: *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering*. 2024, pp. 1–13.

[46] Qianou Ma, Tongshuang Wu, and Kenneth Koedinger. "Is AI the better programming partner? Human-Human pair programming vs. Human-AI pAIr programming". In: *arXiv preprint arXiv:2306.05153* (2023).

[47] Shantanu Mandal et al. "Large language models based automatic synthesis of software specifications". In: *arXiv preprint arXiv:2304.09181* (2023).

[48] Antonio Mastropaolo, Luca Pascarella, and Gabriele Bavota. "Using deep learning to generate complete log statements". In: *Proceedings of the 44th International Conference on Software Engineering*. 2022, pp. 2279–2290.

[49] Qianru Meng et al. "Combining Retrieval and Classification: Balancing Efficiency and Accuracy in Duplicate Bug Report Detection". In: *arXiv preprint arXiv:2404.14877* (2024).

[50] Ambarish Moharil and Arpit Sharma. "Identification of intra-domain ambiguity using transformer-based machine learning". In: *Proceedings of the 1st International Workshop on Natural Language-based Software Engineering*. 2022, pp. 51–58.

[51] Ambarish Moharil and Arpit Sharma. "Tabasco: A transformer based contextualization toolkit". In: *Science of Computer Programming* 230 (2023), p. 102994.

[52] Quim Motger et al. "T-frex: A transformer-based feature extraction method from mobile app reviews". In: *2024 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE. 2024, pp. 227–238.

[53] Manisha Mukherjee and Vincent J Hellendoorn. "Stack over-flowing with results: the case for domain-specific pre-training over one-size-fits-all models". In: *arXiv preprint arXiv:2306.03268* (2023).

[54] Shuyin Ouyang et al. "LLM is Like a Box of Chocolates: the Non-determinism of ChatGPT in Code Generation". In: *arXiv preprint arXiv:2308.02828* (2023).

[55] Bhargavi Paranjape et al. "Art: Automatic multi-step reasoning and tool-use for large language models". In: *arXiv preprint arXiv:2303.09014* (2023).

[56] Amrit Poudel, Jinfeng Lin, and Jane Cleland-Huang. "Leveraging Transformer-based Language Models to Automate Requirements Satisfaction Assessment". In: *arXiv preprint arXiv:2312.04463* (2023).

[57] Julian Aron Prenner and Romain Robbes. "Making the most of small Software Engineering datasets with modern machine learning". In: *IEEE Transactions on Software Engineering* 48.12 (2021), pp. 5050–5067.

[58] Chen Qian et al. "Communicative agents for software development". In: *arXiv preprint arXiv:2307.07924* 6 (2023).

[59] Brian Randell. "The 1968/69 nato software engineering reports". In: *History of software engineering* 37 (1996).

[60] Krishna Ronanki, Beatriz Cabrero-Daniel, and Christian Berger. "ChatGPT as a Tool for User Story Quality Evaluation: Trustworthy Out of the Box?" In: *International Conference on Agile Software Development*. Springer. 2022, pp. 173–181.

[61] Fardin Ahsan Sakib, Saadat Hasan Khan, and AHM Karim. "Extending the frontier of chatgpt: Code generation and debugging". In: *arXiv preprint arXiv:2307.08260* (2023).

[62] Imanol Schlag et al. "Large language model programs". In: *arXiv preprint arXiv:2305.05364* (2023).

[63] Martin Schroder. "Autoscrum: Automating project planning using large language models". In: *arXiv preprint arXiv:2306.03197* (2023).

[64] Ying Sheng et al. "Flexgen: High-throughput generative inference of large language models with a single gpu". In: *International Conference on Machine Learning*. PMLR. 2023, pp. 31094–31116.

[65] Mohammed Latif Siddiq, Beatrice Casey, and Joanna Santos. "A lightweight framework for high-quality code generation". In: *arXiv preprint arXiv:2307.08220* (2023).

[66] Mohammed Latif Siddiq et al. "Exploring the effectiveness of large language models in generating unit tests". In: *arXiv preprint arXiv:2305.00418* (2023).

[67] Ian Sommerville. "Software Engineering". In: 9th ed. Publisher Name, 2011, pp. 5–13.

[68] Giriprasad Sridhara, Sourav Mazumdar, et al. "Chatgpt: A study on its utility for ubiquitous software engineering tasks". In: *arXiv preprint arXiv:2305.16837* (2023).

[69] Yutian Tang et al. "Chatgpt vs sbst: A comparative assessment of unit test suite generation". In: *IEEE Transactions on Software Engineering* (2024).

[70] Shailja Thakur et al. "Verigen: A large language model for verilog code generation". In: *ACM Transactions on Design Automation of Electronic Systems* 29.3 (2024), pp. 1–31.

[71] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[72] Jesse Vig. "A multiscale visualization of attention in the transformer model". In: *arXiv preprint arXiv:1906.05714* (2019).

[73] Julian Von der Mosel, Alexander Trautsch, and Steffen Herbold. "On the validity of pre-trained transformers for natural language processing in the software engineering domain". In: *IEEE Transactions on Software Engineering* 49.4 (2022), pp. 1487–1507.

[74] Junjie Wang et al. "Software testing with large language models: Survey, landscape, and vision". In: *IEEE Transactions on Software Engineering* (2024).

[75] Xingyao Wang et al. "Leti: Learning to generate from textual interactions". In: *arXiv preprint arXiv:2305.10314* (2023).

[76] Yawen Wang et al. "A deep context-wise method for coreference detection in natural language requirements". In: *2020 IEEE 28th International Requirements Engineering Conference (RE)*. IEEE. 2020, pp. 180–191.

[77] Yawen Wang et al. “Where is your app frustrating users?” In: *Proceedings of the 44th International Conference on Software Engineering*. 2022, pp. 2427–2439.

[78] Danning Xie et al. “Impact of large language models on generating software specifications”. In: *arXiv preprint arXiv:2306.03324* (2023).

[79] Zhuokui Xie et al. “ChatUniTest: a ChatGPT-based automated unit test generation tool”. In: *arXiv preprint arXiv:2305.04764* (2023).

[80] Junjielong Xu et al. “UniLog: Automatic Logging via LLM and In-Context Learning”. In: *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering*. 2024, pp. 1–12.

[81] Chengran Yang et al. “Aspect-based api review classification: How far can pre-trained transformer model go?” In: *2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE. 2022, pp. 385–395.

[82] Chengran Yang et al. “Apidocbooster: An extract-then-abstract framework leveraging large language models for augmenting api documentation”. In: *arXiv preprint arXiv:2312.10934* (2023).

[83] Catherine Yeh et al. “Attentionviz: A global view of transformer attention”. In: *IEEE Transactions on Visualization and Computer Graphics* (2023).

[84] Siyu Yu et al. “Log Parsing with Generalization Ability under New Log Types”. In: *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2023, pp. 425–437.

[85] Simiao Zhang et al. “Experimenting a New Programming Practice with LLMs”. In: *arXiv preprint arXiv:2401.01062* (2024).

[86] Ting Zhang et al. “Sentiment analysis for software engineering: How far can pre-trained transformer models go?” In: *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE. 2020, pp. 70–80.

[87] Ting Zhang et al. “Revisiting sentiment analysis for software engineering in the era of large language models”. In: *arXiv preprint arXiv:2310.11113* (2023).

[88] James Xu Zhao et al. “Automatic model selection with large language models for reasoning”. In: *arXiv preprint arXiv:2305.14333* (2023).

[89] Zibin Zheng et al. “Towards an understanding of large language models in software engineering tasks”. In: *arXiv preprint arXiv:2308.11396* (2023).

[90] Wenxuan Zhou et al. “Universalner: Targeted distillation from large language models for open named entity recognition”. In: *arXiv preprint arXiv:2308.03279* (2023).

[91] Yongchao Zhou et al. “Large language models are human-level prompt engineers”. In: *arXiv preprint arXiv:2211.01910* (2022).

[92] Jianfei Zhu et al. “Enhancing traceability link recovery with unlabeled data”. In: *2022 IEEE 33rd International Symposium on Software Reliability Engineering (ISSRE)*. IEEE. 2022, pp. 446–457.

[93] Daniel M Ziegler et al. “Fine-tuning language models from human preferences”. In: *arXiv preprint arXiv:1909.08593* (2019).

# A Promts for categorization

## A.1 System Prompt for Research Focus Categorization

The following prompt is used for categorizing papers into the defined research focus categories using an LLM:

Listing 1: Python code example for extraction chain setup

```
def setup_extraction_chain(self):
        system_prompt = ChatPromptTemplate.from_messages([
                (
                    "system",
                    "You are a highly detailed and context-aware extraction algorithm. "
                    "Your task is to meticulously analyze research papers on large language models (LLMs) for software engineering tasks, focusing on the titles and abstracts. "
                    "Your primary objectives are to categorize the papers based on their research focus and intended goals. "

                    "For the research_focus field, provide the following structure:\n"
                    "research_focus:\n"
                    "  category: [MUST be exactly one of: 'direct_solution', 'dataset_benchmark', 'llm_property_investigation', 'existing_solution_evaluation', 'other' - all lowercase]\n"
                    "  category_reasoning: [Provide a brief explanation for the chosen category]\n"

                    "The categories are defined as follows:\n"
                    "- DIRECT_SOLUTION: Papers proposing a solution Y for Problem(s) X related to SE-Task(s) X. Artifacts may include new frameworks,
```

```
                    concepts, or modifications of
                    existing solutions.\n"
                "- DATASET_BENCHMARK: Papers creating
                    new Datasets or Benchmarks Y'
                    related to Problem(s) X of SE-Task(
                    s) X for assessing the quality of
                    Solutions Y.\n"
                "- LLM_PROPERTY_INVESTIGATION: Papers
                    investigating properties of LLMs
                    related to Problem(s) X of SE-Task(
                    s) X, providing new findings about
                    specific LLM attributes.\n"
                "- EXISTING_SOLUTION_EVALUATION:
                    Papers investigating existing
                    Solution Y's quality for Problem(s)
                     X related to SE-Task(s) X,
                    evaluating performance of existing
                    LLM-based solutions.\n"
                "- OTHER: Papers that do not clearly
                    fit into the above categories.\n"

                "If you choose OTHER, provide a
                    matching new category suggestion in
                     the category_reasoning field.\n"


                "Your analysis should help categorize
                    each paper's research focus
                    accurately, providing a
                    comprehensive understanding of
                    their contributions to the field of
                     software engineering using LLMs."
            ),
            ("human", "{text}"),
        ])

    api_key = OPENAI_KEY
    #llm = ChatOpenAI(model="gpt-3.5-turbo", api_key=
        api_key, temperature=0)gpt-4o
    llm = ChatOpenAI(model="gpt-4o", api_key=api_key,
        temperature=0)
```

## A.2 Extraction Schema for Categorization

The following code defines the schema for categorizing papers based on their research focus:

```
from enum import Enum
from typing import Optional, List
from langchain_core.pydantic_v1 import BaseModel, Field

```

```python
class ResearchFocusCategory(Enum):
    """
    This class defines the categories for classifying research papers based on their focus and intended goals in the context of utilizing LLMs for Software Engineering tasks.
    """
    DIRECT_SOLUTION = 'direct_solution', 'Papers proposing a solution Y for Problem(s) X related to SE-Task(s) X. Artifacts may include new frameworks, concepts, or modifications of existing solutions.'
    DATASET_BENCHMARK = 'dataset_benchmark', 'Papers creating new Datasets or Benchmarks Y\' related to Problem(s) X of SE-Task(s) X for assessing the quality of Solutions Y.'
    LLM_PROPERTY_INVESTIGATION = 'llm_property_investigation', 'Papers investigating properties of LLMs related to Problem(s) X of SE-Task(s) X, providing new findings about specific LLM attributes.'
    EXISTING_SOLUTION_EVALUATION = 'existing_solution_evaluation', 'Papers investigating existing Solution (e.g. ChatGPT or Codex) Y\'s quality for Problem(s) X related to SE-Task(s) X, evaluating performance of existing LLM-based solutions.'
    OTHER = 'other', 'Papers that do not clearly fit into the above categories.'

    def __new__(cls, value, description):
        member = object.__new__(cls)
        member._value_ = value
        member.description = description
        return member

class ResearchFocus(BaseModel):
    category: List[Optional[ResearchFocusCategory]] = Field(
        default=None,
        description="Classification of the paper's research focus and intended goal"
    )
    category_reasoning: List[Optional[str]] = Field(
        default=None,
        description="Reasoning for the classification. If classified as OTHER, provide a matching new category suggestion here."
    )

class ArticleMetadata(BaseModel):
    """Extract information from paper's title and abstract """
```

```
33      research_focus: Optional[ResearchFocus] = Field(
34          default=None,
35          description="Classification of the paper's
                research focus and intended goal"
36      )
```