Revisiting Sentiment Analysis for Software Engineering in the Era of Large Language Models

TING ZHANG, Singapore Management University, Singapore
IVANA CLAIRINE IRSAN, Singapore Management University, Singapore
FERDIAN THUNG, Singapore Management University, Singapore
DAVID LO, Singapore Management University, Singapore

Software development involves collaborative interactions where stakeholders express opinions across various platforms. Recognizing the sentiments conveyed in these interactions is crucial for the effective development and ongoing maintenance of software systems. For software products, analyzing the sentiment of user feedback, e.g., reviews, comments, and forum posts can provide valuable insights into user satisfaction and areas for improvement. This can guide the development of future updates and features. However, accurately identifying sentiments in software engineering datasets remains challenging.

This study investigates bigger large language models (bLLMs) in addressing the labeled data shortage that hampers fine-tuned smaller large language models (sLLMs) in software engineering tasks. We conduct a comprehensive empirical study using five established datasets to assess three open-source bLLMs in zero-shot and few-shot scenarios. Additionally, we compare them with fine-tuned sLLMs, using sLLMs to learn contextual embeddings of text from software platforms.

Our experimental findings demonstrate that bLLMs exhibit state-of-the-art performance on datasets marked by limited training data and imbalanced distributions. bLLMs can also achieve excellent performance under a zero-shot setting. However, when ample training data is available or the dataset exhibits a more balanced distribution, fine-tuned sLLMs can still achieve superior results.

CCS Concepts: • **Software and its engineering** → **Maintaining software**.

Additional Key Words and Phrases: Large Language Models, Sentiment Analysis, Software Engineering

1 INTRODUCTION

Sentiment analysis (SA), or opinion mining, is the computational study of people's opinions or emotions toward entities [34]. SA holds significant practical value and has found diverse applications across a wide spectrum of domains, including but not limited to business, marketing, politics, healthcare, and public advocacy [16]. Generally speaking, SA contains several tasks, such as sentiment classification [32], aspect-based sentiment classification [73], or hate speech detection [53]. Most existing Software Engineering (SE) research focuses on sentiment classification, i.e., assigning a sentiment polarity (e.g., negative, neutral, and positive) to a given text unit. For simplicity, we refer to the sentiment classification as SA in the rest of this article. SA has proven its utility in various SE tasks, exemplified by its role in evaluating user reviews of mobile applications [31] and identifying sentences conveying negative opinions about application programming interfaces

(APIs) [74]. Given that SE is inherently a collaborative endeavor, comprehending the sentiments expressed by different stakeholders across various platforms becomes imperative for the effective development and maintenance of software systems.

Prior studies have demonstrated that general SA tools work well on social media posts or product reviews while performing poorly on SE datasets [27, 58]. This discrepancy has spurred a growing interest in developing SE-specific SA tools over the past decade [1, 10, 26]. These sentiment analysis for SE (SA4SE) tools usually either propose a SE-specific lexicon [26] or a SE-specific model [1]. At the same time, several benchmarking studies on evaluating general SA tools and SE-specific tools have been conducted [27, 31, 43, 71]. Zhang et al. [71] made the first attempt to embrace the power of language models, i.e., small pre-trained large language models (sLLMs),[1] for SA4SE. Zhang et al. demonstrated that sLLMs outperform existing specialized SA4SE tools on the evaluation datasets. However, several challenges persist in the field of SA4SE. First, the accuracy of sLLMs can degrade when there is a lack of labeled data for fine-tuning. For instance, the Google Play dataset [31], which contains app review comments, only has 341 labeled documents and among them 25 are neutral ones. The fine-tuned sLLMs predicted none of the data points in the test set as neutral. While acquiring more labeled data can help mitigate this issue, manually labeling large volumes of data is time-consuming. The second challenge relates to the limitations of fine-tuning itself. Fine-tuning sLLMs requires updating some of the model parameters with domain-specific data. Lastly, the third challenge occurs in cross-platform settings [41], where SA4SE tools tend to perform poorly. Models trained from one dataset may not generalize well when tested on a different dataset, hindering the generalizability and effectiveness of existing SA4SE tools. Given these challenges, there is a need to explore more effective solutions for SA4SE.

Recently, large language models have shown promising results in many areas, spanning from general natural language processing (NLP) tasks to specialized applications like software development. bLLMs are usually trained on massive corpora of texts and contain many parameters. For instance, GPT-3 [6] contains 175 billion parameters. LLAMA [59] is trained on trillions of tokens and contains 7B to 65B parameters. Given the large number of parameters, fine-tuning bLLMs for every downstream task is impractical. These bLLMs permit *in-context learning*: they can be adapted to a downstream task simply by providing it with a prompt (a natural language description of the task) [4]. This adaptability has been a game-changer in reducing the need for domain-specific training data, as bLLMs can leverage their pre-existing knowledge to excel in diverse applications. In-context learning has drastically reduced the domain-specific training examples required for a particular application [13]. bLLMs can make predictions conditioned on a few input-output examples without updating model parameters and achieve success in various tasks [6, 13]. Nevertheless, their performance in SA4SE remains largely unexplored. The intriguing prospect of adopting bLLMs in this context lies in their ability to potentially address the challenges associated with fine-tuning sLLMs and the limitations observed in cross-platform settings.

To fill this gap, our work embarks on a journey to explore the effectiveness of bLLMs for SA4SE. To investigate the effectiveness of bLLMs for SA4SE, we conducted a comprehensive empirical study on five existing SE datasets. We first evaluate bLLMs under zero-shot and few-shot settings. For the zero-shot setting, we experimented with three different prompt templates. For the few-shot setting, we experimented with 1-, 3-, and 5-shot. The experimental results demonstrate that bLLMs can perform well under a zero-shot setting, while few-shot learning can further boost the performance. However, adding more shots does not guarantee an improvement in the performance. We also

---

[1]To distinguish from the recent larger sizes of large language models, we consider **sLLMs** as relatively smaller sizes of large language models that can be easily fine-tuned locally, such as BERT, RoBERTa, and XLNet. sLLMs usually contain <1B parameters. We refer to the bigger large language models, which contain billions of parameters as **bLLMs**.

compared prompting bLLMs with fine-tuning sLLMs. We find that, on the dataset that lacks training data and the data is highly imbalanced, bLLMs can surpass sLLMs by a large gap. For the datasets that contain sufficient training data and more balanced data, sLLMs may still be preferred.

Our contribution can be summarized as follows:

- Our work is the first study to examine the effectiveness of open-source bLLMs on the SA4SE task.
- We evaluate three open-source bLLMs under zero and few-shot settings.
- We compare fine-tuned sLLMs with bLLMs on five SE datasets collected from five distinct platforms.

The remaining parts of this work are as follows: Section 2 introduces the background of our work. Section 3 discusses about our experimental setup. Section 4 presents the results of our empirical study. Section 5 discusses the additional experiments, the implications of our findings, and the threats to validity. Section 6 concludes this work and discusses future work.

2 RELATED WORK

In recent years, SA4SE has emerged as a vibrant and active research area within SE. LLMs also have been widely applied in the SE domain. This section provides an overview of the relevant literature, primarily focusing on the techniques proposed to boost SA4SE accuracy and empirical studies in SA4SE.

2.1 Boosting SA4SE Accuracy

In the past decades, many techniques have been proposed to improve the effectiveness of identifying sentiments or emotions in the SE domain [1, 3, 7, 8, 10, 11, 22, 24–26, 40].

Chen et al. [10] propose SEntiMoji, an emoji-powered learning approach for SA in SE. They employ emotional emojis as noisy labels of sentiments and propose a representation-learning approach that uses both Tweets and GitHub posts containing emojis to learn sentiment-aware representations for SE-related texts. In the evaluation, they compare SEntiMoji with four SA4SE tools on sentiment polarity benchmark datasets. The experimental results show that SEntiMoji can significantly improve the performance.

Furthermore, Chen et al. [11] include an additional evaluation of SEntiMoji on the emotion detection task. They also compared it with four existing emotion detection methods, including DEVA [25], EmoTxt [8], MarValous [24], and ESEM-E [40]. The experimental results on the five benchmark datasets covering 10,096 samples for sentiment detection and four benchmark datasets covering 10,595 samples for emotion detection demonstrate that SEntiMoji is effective.

Besides developing a new SA4SE tool, another research line aims at improving SA accuracy by handling existing challenges, such as labeled data scarcity. Imran et al. [22] address the data scarcity problem by automatically creating new training data using a data augmentation technique. They specifically target the data augmentation strategy to improve the performance of emotion recognition by analyzing the types of errors made by popular SE-specific emotion recognition tools. Their results show that when trained with their best augmentation strategy, three existing emotion classification tools, i.e., ESEM-E, EMTk, and SEntiMoji, received an average improvement of 9.3% in micro-F1 score.

As previously discussed in Section 1, Zhang et al. [71] introduced the approach of fine-tuning sLLMs for SA4SE. Our approach distinguishes itself from the aforementioned research by pioneering the use of prompting bLLMs for SA4SE. Alongside these efforts, we are collectively working towards advancing the accurate identification of sentiment within the SE field.

2.2 Empirical Studies in SA4SE

With the proliferation of domain-specific SA4SE tools, a series of empirical investigations have been conducted to illuminate our understanding of this field's progress and challenges [27, 28, 30, 31, 42–44].

Novielli et al., in their recent study [42], delve into the critical question of how off-the-shelf, SE-specific SA tools affect the conclusion validity of empirical studies in SE. They begin by replicating two prior studies that explore the role of sentiment in security discussions on GitHub and question-writing on Stack Overflow. Subsequently, they extend these studies by assessing the level of agreement between different SA tools and manual annotations, using a gold standard dataset comprising 600 documents. The experimental findings from this research reveal that when applied out-of-the-box, various SA4SE tools may yield conflicting results at a granular level. Consequently, it becomes imperative to consider platform-specific fine-tuning or retraining to account for differences in platform conventions, jargon, or document lengths.

Obaidi et al. [44] conducted a systematic mapping study to comprehensively examine SA tools developed for or applied in the SE domain. This study summarizes insights drawn from 106 papers published up to December 2020, focusing on six key aspects: (1) the application domain, (2) the purpose of SA, (3) the datasets used, (4) the approaches for developing SA tools, (5) the utilization of pre-existing tools, and (6) the challenges faced by researchers. Based on their findings, neural networks emerge as the top-performing approach, with BERT identified as the most effective tool.

Beyond the scope of sentiment classification, some researchers have explored broader facets of SA4SE, such as opinion mining. Opinion mining encompasses a wider spectrum of tasks than the sentiment polarity identification typically evaluated in SA4SE studies. It includes SA, subjectivity detection, opinion identification, and joint topic SA. In a comprehensive systematic literature review, Lin et al. investigated 185 papers on opinion mining in Software Engineering [30], shedding light on the diverse research efforts in this area.

Each of these existing empirical studies has contributed valuable insights into the evolving landscape of SA4SE. Notably, our work stands apart from these studies as it introduces the use of bLLMs to this domain for the first time.

3 EXPERIMENTAL SETUP

3.1 Research Questions

In this work, we plan to answer the following research questions (RQs):

RQ1: **How do various prompts affect the performance of bLLMs in zero-shot learning for the SA4SE task?**
In this RQ, our initial focus is exploring the zero-shot learning scenario, where bLLMs are prompted without providing any labeled data. Prior studies have unveiled that bLLMs exhibit varying results even when prompted with semantically similar queries [36, 39, 47]. Additionally, earlier research findings have emphasized the substantial impact of different word orders within the prompt templates on the predictions [14, 38]. Nevertheless, it is yet to be determined if bLLMs exhibit varying performance when prompted with templates that have subtle differences on the SA4SE task. We experiment with slight variations of the prompts to precisely measure the bLLMs' sensitivity to nuanced linguistic cues. This approach is essential in controlled experimental settings where isolating the impact of single variables is crucial. By altering the prompts minimally, we can more accurately attribute any observed differences in performance to these specific changes, thereby gaining insight into how the model processes and interprets instructions at a granular level. Additionally, we believe the commands used to conduct SA are rather straightforward, making it challenging to create two
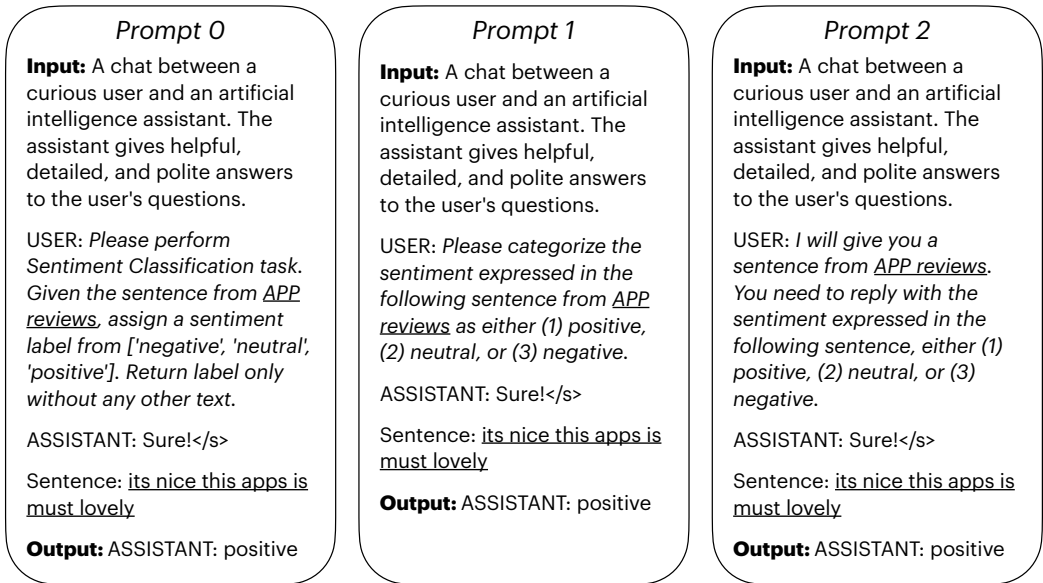
Fig. 1. The zero-shot prompt templates we utilized when running VICUNA [12] and WIZARDLM [64]

prompts with substantial changes. To evaluate the effectiveness of different prompt templates in generating accurate predictions, we hypothesize that there are significant differences in performance metrics (macro-F1, micro-F1, and AUC scores) among the various prompt templates. Specifically, we propose the following hypotheses for pairwise comparisons:

- Prompt 0 vs. Prompt 1
  * H0.1.1 (Null Hypothesis): There is no significant difference in the performance metrics between Prompt 0 and Prompt 1.
  * H1.1.1 (Alternative Hypothesis): There is a significant difference in the performance metrics between Prompt 0 and Prompt 1.
- Prompt 0 vs. Prompt 2
  * H0.1.2 (Null Hypothesis): There is no significant difference in the performance metrics between Prompt 0 and Prompt 2.
  * H1.1.2 (Alternative Hypothesis): There is a significant difference in the performance metrics between Prompt 0 and Prompt 2.
- Prompt 1 vs. Prompt 2
  * H0.1.3 (Null Hypothesis): There is no significant difference in the performance metrics between Prompt 1 and Prompt 2.
  * H1.1.3 (Alternative Hypothesis): There is a significant difference in the performance metrics between Prompt 1 and Prompt 2.

We will employ a Wilcoxon signed-rank test with Bonferroni correction for multiple comparisons to assess the significance of the differences between these prompt templates. The corrected alpha threshold for significance will be set at 0.0167 to account for the multiple pairwise comparisons.

RQ2: **How do various shots affect the performance of bLLMs in few-shot learning for the SA4SE task?**

In this RQ, we evaluate whether bLLMs exhibit enhanced performance in scenarios with a

> **Input:** <s>[INST] <<SYS>>
>
> A chat between a curious user and an artificial intelligence assistant. The assistant gives helpful, detailed, and polite answers to the user's questions.
>
> <</SYS>>
>
> Please perform Sentiment Classification task. Given the sentence from APP reviews, assign a sentiment label from ['negative', 'neutral', 'positive']. Return label only without any other text.
>
> Example Sentence: this app is a waste of time. i can log in but it always freezes and eventually crashes. i do not feel confident with the security on this app if wf can leave us hanging with a horrible app for such a big bank. uninstalled htc one
>
> Label: negative   Demonstration
>
> [/INST]
>
> Sure!
>
> </s><s>[INST]
>
> Sentence: good it's good app
>
> Label:
>
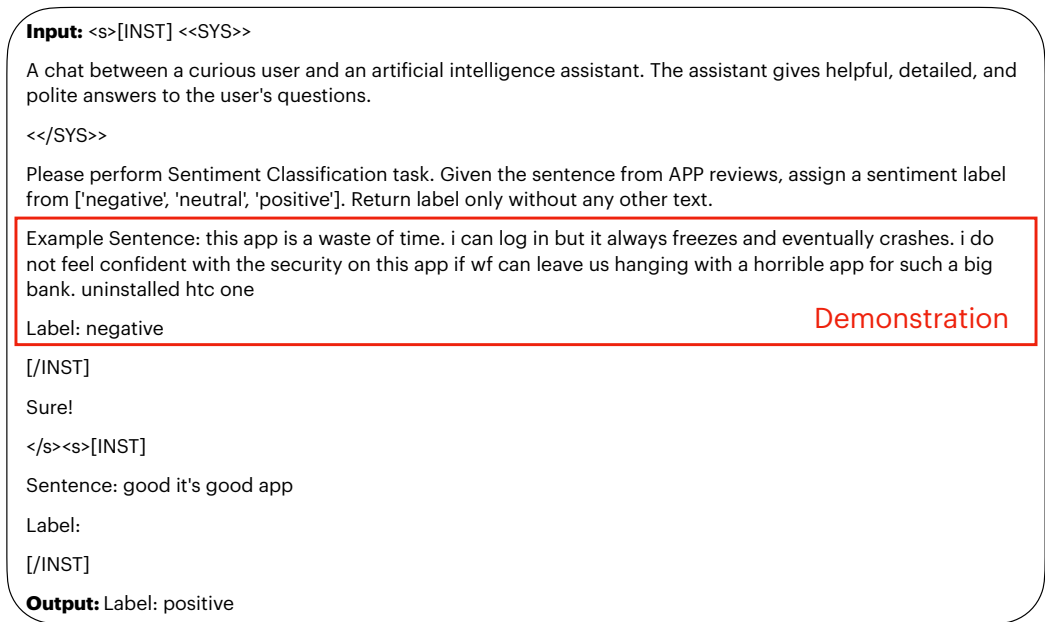> [/INST]
>
> **Output:** Label: positive

Fig. 2. Few-shot prompt template (with $k = 1$) utilized by LLAMA 2-CHAT [60].

limited number of examples available. In contrast to zero-shot learning, few-shot learning incorporates an extra "Demonstration" component. While previous studies, such as Zhang et al. [72], indicate that few-shot learning may surpass zero-shot learning in certain aspects, contrasting findings by Reynolds et al. [49] suggest that zero-shot prompts can significantly outperform few-shot prompts in some NLP tasks. Nonetheless, it is still unclear whether bLLMs derive greater benefits from few-shot learning, specifically in the SA4SE task. We aim to test the following null hypothesis: H0.2: there is no significant difference in the results obtained under few-shot learning compared to those from zero-shot learning.

RQ3: **How do prompting bLLMs compare with fine-tuned sLLMs on the SA4SE task?**
Prior work has shown that fine-tuning sLLMs can achieve state-of-the-art results on the SA4SE task. However, the effectiveness of prompting bLLMs compared to fine-tuning sLLMs remains an open question. To address this, we compare the best macro-F1 and micro-F1 scores achieved through prompting bLLMs with those from fine-tuned sLLMs. To further explore this comparison, we test the null hypothesis: H0.3: There is no significant difference in results obtained from prompting bLLMs and fine-tuning sLLMs.

RQ4: **What are the factors leading to the misclassification of sentiment labels by bLLMs?**
To gain a deeper understanding of the reasons behind incorrect sentiment label classifications by bLLMs in the context of software engineering, we intend to carry out a detailed manual analysis.

3.2 Method

In this section, we present the steps performed to answer our research questions.

For **RQ1**, we investigate how various prompt templates affect the results. Given the straightforward nature of SA, our objective is to formulate equally straightforward prompts. These prompts

encompass two key components: the *Task Description* and the *Output Format*. The *Task Description* serves the purpose of elucidating the task clearly and concisely. In our specific context, the task pertains to SA, and we articulate it through various expressions within the three templates. Importantly, the sentence origin (e.g., from APP reviews, from Stack Overflow) remains consistent across all prompt templates, enabling the distinction of diverse contexts and domains. The *Output Format* component is designed to provide bLLMs with guidance for generating responses in a specific format, facilitating sentiment label extraction. To maintain generality, we employ an identical prompt template for all five datasets.[2] Figure 1 shows the three prompt templates we used for zero-shot setting. While all prompt templates share a semantic similarity, they differ in their syntactic structure.

Our inspiration for the first prompt template (i.e., Prompt 0) draws from Zhang et al. [72]. They designed the prompt to include only essential components, namely the task name, task definition, and output format. In addition to these components, considering the specific context of SE which differs from a general context, we have incorporated the sentence origin. This addition aims to enrich the context provided to the bLLMs, enhancing their effectiveness in this specialized field. For Prompt 1 and Prompt 2, we intentionally implemented subtle variations, focusing specifically on the aspects of *structure* and *tone*. Regarding the *structure* of the prompts, Prompt 0 stands out as the most formal and task-oriented. In contrast, Prompts 1 and 2 introduce sentiment categories in an option-based format (1, 2, 3), rather than a straightforward list. This choice gives these prompts a more interactive and engaging feel, making them especially suitable for situations that require user interaction. In addition, Prompt 2 presents a more interactive scenario where the user is actively involved ("I will give you a sentence..."). This format implies a back-and-forth interaction. In terms of *tone*, Prompt 1 is slightly more engaging than Prompt 0, as it reads more like a request than a command. Prompt 2 further amplifies this interactive quality by establishing a dialogue-like contex.

For **RQ2**, in the context of few-shot learning, we leverage the best-performing zero-shot prompt template, namely *Prompt 0*. We enrich *Prompt 0* with various numbers of examples filled in the **Demonstration** part. The demonstration part encompasses k ($k = 1, 3, 5$) examples and corresponding ground-truth labels, adhering to the desired format. Figure 2 illustrates the few-shot prompt template employed by LLAMA 2-CHAT on the GooglePlay dataset. In the depicted figure, the demonstration segment (enclosed within the red box) comprises only one example. In the case of a 3-shot or 5-shot setup, this demonstration section would encompass a greater number of example sentences and their corresponding gold labels. We systematically sampled 1-, 3-, and 5-examples from the training data of each dataset, subsequently populating the demonstration segment within the template. This approach ensures that under the k-shot setting, different bLLMs receive the same set of examples.

For **RQ3**, we aim to compare prompting bLLMs and fine-tuning sLLMs. Thus, we ensure that both bLLMs and sLLMs make predictions on the same test data set. We report the highest macro- and micro-F1 scores achieved by the bLLM on the dataset as the result achieved by the bLLM.

In **RQ4**, we conduct a quantitative and qualitative analysis to understand the main cause of misclassification made by bLLMs. For the quantitative aspect, we focus on the results from the most effective templates. Specifically, we analyze the results from *Prompt 0* for zero-shot learning and from the *5-shot* prompt for few-shot learning. In the qualitative part of our study, we utilize the error categorization framework established by Novielli et al. [41, 43]. Employing these predefined categories, our analysis involved two authors (from here on, evaluators) independently assigning a

---

[2]There is only a minor difference in the Jira dataset, given it only contains two sentiments, i.e., negative and positive. We reduced the scope to only two options in the templates used by Jira.

category to each document in our study. It is important to note that not all categories identified by Novielli et al. were necessarily covered in each document. Subsequently, the evaluators engaged in discussions to resolve any conflict and reach a consensus.

3.3 Dataset

In this work, we experimented with the existing manually labeled datasets from five distinct platforms: Gerrit, GitHub, Google Play, Jira, and Stack Overflow. For simplicity, we refer to these datasets using abbreviations: Gerrit [1], GitHub [41], GooglePlay [31], Jira [31], and StackOverflow [31].

Gerrit Dataset: Ahmed et al. [1] meticulously labeled this dataset. They initiated their process by mining code review repositories from 20 prominent open-source software (OSS) projects. Three raters individually labeled the selected code review comments and resolved conflicts through discussion. The dataset was refined into two classes: *negative* and *non-negative*, forming the final dataset.

GitHub Dataset: Novielli et al. [41] curated the GitHub dataset, which comprises pull request and commit comments. Sentiment was assessed based on the entire comment, rather than isolated portions. The labeling process began with the manual classification of 4,000 comments, followed by a semi-automatic approach using Senti4SD [7], which required manual confirmation of the automatically assigned polarity labels.

GooglePlay, Jira, and StackOverflow Datasets: Lin et al. [31] provided three datasets, each with its unique characteristics:

- **GooglePlay Dataset**: Originally collected by Chen et al. [9], this dataset contains user reviews of Android apps on Google Play. Villarroel et al. [62] selected a subset of reviews from Chen et al.'s dataset, and Lin et al. further sampled 341 reviews. Lin et al. performed the manual labeling of sentiment, where two annotators individually classified text as positive, neutral, or negative. In cases of disagreement, a third evaluator was involved for resolution.
- **Jira Dataset**: This dataset comprises Jira issue comment sentences and was originally collected and labeled by Ortu et al. [45]. However, Ortu et al.'s dataset only provided emotional labels, such as *love*, *joy*, *anger*, and *sadness*. Lin et al. mapped sentences labeled with *love* or *joy* to "positive" and those labeled with *anger* or *sadness* to "negative".
- **StackOverflow Dataset**: Lin et al. gathered and labeled the StackOverflow dataset, extracting 5,073,452 sentences from the latest Stack Overflow dump available in July 2017. The sentences were selected based on two criteria: they had to be tagged with "Java", and they needed to contain keywords such as "library/libraries" or "API(s)". A random sample of 1,500 sentences was manually labeled by assigning a sentiment score to each sentence. Two annotators carried out the manual annotation individually, with conflicts resolved through discussion.

We split each dataset with a ratio of 8:1:1, which stands for training, validation, and test, respectively. We did a stratified split where we kept the original class distribution in training, validation, and test. Since running bLLMs is expensive, they usually contain billions of parameters, we did a sampling on all the test data with a confidence level of 95% and a margin of error of +/- 5%.[3] Similarly, we also kept the class distribution the same as in the original whole dataset.

Table 1 presents the statistics of investigated datasets, specifically the average number of tokens per document. Notably, Gerrit and GooglePlay exhibit longer text, likely due to the inclusion

---

[3]We included all the provided test data in the GooglePlay dataset, as the number of sampled data is only 2 data points fewer than the whole test data.

Table 1. Dataset statistics. Neg. stands for negative, Neu. stands for neutral, Pos. stands for positive, and Non-neg. stands for non-negative.

Dataset	Total	Test	Sampled Test	Avg. Tokens
Gerrit	1,600: 398 (Neg.) 1,202 (Non-neg.)	160	114	29
GitHub	7,122: 2,087 (Neg.) 3,022 (Neu.) 2,013 (Pos.)	713	250	19
GooglePlay	341: 130 (Neg.) 25 (Neu.) 186 (Pos.)	35	35	27
Jira	926: 636 (Neg.) 290 (Pos.)	93	76	9
StackOverflow	1,500: 178 (Neg.) 1,191 (Neu.) 131 (Pos.)	150	109	11

of code review and APP review comments, which often span multiple sentences. Although the GitHub dataset also comprises pull request and commit comments, they are typically short in nature. Conversely, the StackOverflow dataset and Jira dataset include *sentences* from Stack Overflow and Jira, respectively. While some document units in these datasets contain multiple sentences and others just one, we collectively refer to them as *documents*. Note that the Gerrit, GooglePlay, Jira, and StackOverflow datasets investigated in our study are imbalanced, while the GitHub dataset is more balanced.

3.4 Evaluated Language Models

bLLMs. We include three recently proposed bLLMs based on their performance in the MMLU benchmark on the chatbot leaderboard [4] in August 2023; the model name in the parenthesis is the exact model variant we used on the Hugging Face platform [63].

- **LLAMA 2-CHAT** (meta-llama/Llama-2-13b-chat-hf) [60] is a fine-tuned version of LLAMA 2 that is optimized for dialogue use cases. LLAMA 2 uses the standard Transformer architecture [61] and it applies pre-normalization with RMSNorm [69], the SwiGLU activation function [54], and rotary positional embeddings [56]. LLAMA 2 made several improvements over LLAMA 1, including but not limited to more robust data cleaning, trained on 40% more total tokens, and doubled the context length.
- **VICUNA** (lmsys/vicuna-13b-v1.5) [12] is a chatbot trained by fine-tuning LLAMA 2 on 70K user-shared ChatGPT conversations. To better handle multi-turn conversations and long sequences, VICUNA is trained with the enhanced training script from Alpaca [57].
- **WIZARDLM** (WizardLM/WizardLM-13B-V1.2) [64] is another fine-tuned version of LLAMA 2. The authors propose *Evol-Instruct*, which is a novel method using bLLMs instead of humans to automatically mass-produce open-domain instructions of various complexity levels to improve the performance of bLLMs. The resulting bLLMs by fine-tuning LLAMA 2 with the evolved instructions is called WIZARDLM.

sLLMs. We include all the four sLLMs evaluated by Zhang et al. [71], i.e., BERT [15], RoBERTa [35], ALBERT [29], XLNet [67]. In addition, we also include a lightweight and memory-efficient variant of BERT, i.e., DistilBERT. We briefly describe these models. They mainly differ in the pre-training tasks adopted. We also present the exact model we used on the Hugging Face platform [63] in the parenthesis.

- **BERT** (bert-base-uncased) [15], which stands for Bidirectional Encoder Representations from Transformers, introduces two key pre-training tasks. The first is mask language modeling

---

[4]https://huggingface.co/spaces/lmsys/chatbot-arena-leaderboard

(MLM), where BERT learns to predict masked words in a given text. Additionally, BERT incorporates the next sentence prediction (NSP) task, training to determine whether the second sentence logically follows the first or is a random sentence from the training data.

- **RoBERTa** (`roberta-base`) [35] is short for "A Robustly Optimized BERT Pretraining Approach". RoBERTa is a BERT variant distinguished by its innovative training strategies and hyperparameter choices. Notably, it eliminates the NSP task, employs a larger batch size, trains on a larger corpus than BERT, and utilizes a dynamic masking strategy during training.
- **ALBERT** (`albert-base-v2`) [29], or "A Lite BERT", is another BERT variant designed to reduce model size and computational requirements while maintaining or improving performance. ALBERT retains the MLM task but replaces the NSP task with the sentence order prediction (SOP) task. In SOP, ALBERT is trained to predict whether pairs of sentences are correctly ordered or if their positions have been swapped.
- **XLNet** (`xlnet-base-cased`) [67] primarily focuses on capturing contextual information and long-range dependencies in text. It employs an autoregressive pretraining method and introduces permutation language modeling, where word order in a sentence is randomly shuffled, and the model is trained to predict the original sequence. XLNet also incorporates innovations such as the "two-stream self-attention" mechanism.
- **DistilBERT** (`distilbert-base-uncased`) [51] is a distilled and smaller version of the BERT model. DistilBERT is designed to be faster and more memory-efficient. DistilBERT adopts model compression or knowledge distillation to learn from a teacher BERT to capture the same knowledge but with fewer parameters. As DistilBERT combines efficiency and strong performance, it has been popular in research and industry settings.

3.5 Evaluation Metrics

Following prior works [41, 43, 71], we also report macro- and micro-averaged precision, recall, and F1-score.

$$Precision = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{1}$$

$$Recall = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2}$$

$$F1 = \frac{2 \times (Precision \times Recall)}{Precision + Recall} \tag{3}$$

$$Macro\text{-}F1 = \frac{F1_{negative} + F1_{neutral} + F1_{positive}}{3} \tag{4}$$

Macro-averaged metrics: To get the macro-averaged metrics, we calculate each class and find their unweighted mean. In our context, if we have three labels, i.e., negative, natural, and positive, we separately calculate the F1 for each class (See Formula 1-3). Macro-F1 would be the average of the F1 for these three classes (See Formula 4).

Micro-averaged metrics: Formulate 5, 6, and 7 shows how to calculate *micro-precision*, *micro-recall*, and *micro-F1*.

$$Micro\text{-}Precision = \frac{\text{Total TP}}{\text{Total TP} + \text{Total FP}} \tag{5}$$

$$Micro\text{-}Recall = \frac{\text{Total TP}}{\text{Total TP} + \text{Total FN}} \tag{6}$$

$$Micro\text{-}F1 = \frac{2 \times (Micro\text{-}Precision \times Micro\text{-}Recall)}{Micro\text{-}Precision + Micro\text{-}Recall} \tag{7}$$

Recall that TP, FP, and FN are short for the true positives (i.e., where the model correctly predicts a positive class, and it matches the ground truth), false positives (i.e., where the model incorrectly predicts a positive class, but the ground truth is actually a negative class), and false negatives (i.e., where the model incorrectly predicts a negative class, but the ground truth is actually a positive class). "Total TP" represents the sum of true positives, "Total FP" represents the sum of false positives, and "Total FN" represents the sum of false negatives over all classes.

We report both the macro-F1 and micro-F1 scores as they show the balance between precision and recall. We attach the full result in our replication package.[5] From these formulas, we can find that micro-F1 emphasizes overall accuracy, while macro-F1 gives equal weight to each class's performance. As we do not give weights to the classes and want to consider the F1 in each class, we choose macro-F1 as the main metric. This choice of preferring macro-F1 also aligns with the prior work [41].

AUC: Other than the widely used F1 score, we also report the area under the ROC curve (AUC) to evaluate the performance of the models. AUC measures the entire two-dimensional area underneath the entire ROC curve. AUC is a popular metric for binary classification tasks, measuring the model's ability to distinguish between positive and negative classes. AUC values range from 0 to 1, with higher values indicating better performance. Some of our datasets have three classes, so we use the one-vs-rest strategy to compute the AUC of each class against the rest [19].

3.6 Implementation Details

Note that we only fine-tune sLLMs and prompt bLLMs, while (1) not fine-tuning bLLMs as bLLMs contain a large number of parameters, it is expensive to fine-tune them; and (2) not prompting sLLMs as they usually are pre-trained with the MLM task mentioned before. They predict missing words in a sentence, which differs from the autoregressive nature of models like GPT-3 that generate text sequentially based on a prompt. This architectural difference makes it less straightforward to use BERT for prompt-style tasks. sLLMs, like BERT, were designed for understanding existing text, not generating new text sequences. For example, if we input the prompt "this app is a waste of time..." (as shown in Figure 2) into BERT, it will produce contextualized representations for each token but will not generate a sentiment token or new text. BERT's pre-training objective focused on representing input text, not text generation. To adapt BERT for sentiment classification, we fine-tune it on a labeled dataset to predict sentiment labels (positive, negative, neutral) based on the input text. During fine-tuning, the model learns to leverage BERT's contextualized representations for accurate predictions but does not learn to generate new sequences. Hence, we fine-tune sLLMs for classification tasks like sentiment analysis, not text generation. In contrast, bLLMs like LLAMA 2-CHAT are pre-trained for text generation by predicting the next token based on context, making them better suited for generating coherent text sequences from prompts.

Prompting bLLMs. We use heuristics to extract the sentiment returned by bLLMs. When bLLMs consider it hard to decide the sentiment (they replied with a sentiment other than the three polarities, e.g., "mixed"), we label the predicted sentiment as "neutral". As there is no "neutral" sentiment in the Jira dataset, if any bLLM predicts the sentiment as "neutral", we label the predicted sentiment

---

[5]https://github.com/soarsmu/LLM4SA4SE

1. **Input:** A chat between a curious user and an artificial intelligence assistant. The assistant gives
2. helpful, detailed, and polite answers to the user's questions.
3. USER: Please perform Sentiment Classification task. Given the sentence from APP reviews,
4. assign a sentiment label from ['negative', 'neutral', 'positive']. Return label only without any other
5. text.
6. ASSISTANT: Sure!</s>
7. USER: Sentence: its nice this apps is must lovely
8. ASSISTANT: positive
9. USER: What are predicted probabilities for each class?
10. ASSISTANT: Probability score for positive:
11. **Output:** 0.8
12. Probability score for negative: 0.1
13. Probability score for neutral: 0.1

Fig. 3. One example to get the prediction probability scores from the bLLMs.

as the opposite of the ground-truth label. In the few-shot setting, we select k examples ("shots") at random from the training set (($k = \{1, 3, 5\}$)), and for each example, we append its ground-truth label. To make a new prediction for a new example, we append one sentence from the test set.

It is important to note that bLLMs do not directly provide probability scores for each class. To obtain these scores, we use the predicted result as a new prompt and ask the model to give the probability scores for each class. For example, Figure 3 illustrates an instance using the VICUNA model with Prompt 0 in a zero-shot setting on the GooglePlay dataset. Lines 1-7 show the original Prompt 0, while Line 8 displays the sentiment predicted by VICUNA. Lines 9-10 are added to the original prompt to request the model to provide probability scores for each class. Lines 11-13 then present the probability scores given by the model. We proceed to extract these probability scores for each class. If the total probability scores do not sum to 1, we normalize the scores by dividing each score by the total sum of the scores.

Fine-tuning sLLMs. We fine-tuned all the sLLMs with the training data. For each epoch, we calculate their macro-F1 score on the validation data. We fine-tune each sLLM 5 epochs. We save the best-performing model, i.e., achieving the highest macro-F1 score on the validation data, as the final model. We then evaluate the best model on the test data. We used the following sets of hyper-parameters for all the sLLM: learning rate of 2e-5, batch size of 32, and max length of 256.

4 RESULTS

4.1 RQ1: Impact of different prompts on the performance of bLLMs with zero-shot learning

It is worth noting that VICUNA and WIZARDLM adopt the same style of prompt template; while LLAMA 2-CHAT employs a different prompt template in pre-training, resulting in slight differences in prompt formats[6] (We show an example of LLAMA 2-CHAT prompt in Figure 2, where the zero-shot template is the same template excluding the "Demonstration" part).

Table 2 presents the outcomes obtained from our investigation into three distinct bLLMs using three distinct zero-shot prompts. Notably, we observe varying performance levels among these bLLMs when employed with different prompt templates. Furthermore, it is noteworthy that even

---

[6]https://huggingface.co/blog/llama2#how-to-prompt-llama-2

Table 2. Zero-shot Performance: Comparative Results of LLMs Across Five Datasets. Cells highlighted in red indicate the highest scores achieved among the three prompts executed by each respective model.

	Model	Variant	Gerrit	GitHub	GooglePlay	Jira	StackOverflow
Macro-F1	Llama 2-Chat	0	0.73	0.68	0.89	0.83	0.45
		1	0.71	0.64	0.89	0.71	0.5
		2	0.75	0.68	0.89	0.78	0.51
	Vicuna	0	0.73	0.72	0.98	0.85	0.59
		1	0.73	0.65	0.74	0.69	0.56
		2	0.7	0.67	0.82	0.75	0.53
	WizardLM	0	0.69	0.71	0.8	0.81	0.41
		1	0.69	0.7	0.82	0.82	0.59
		2	0.68	0.7	0.79	0.77	0.52
Micro-F1	Llama 2-Chat	0	0.82	0.68	0.91	0.84	0.61
		1	0.82	0.64	0.91	0.71	0.72
		2	0.83	0.68	0.94	0.79	0.64
	Vicuna	0	0.81	0.72	0.97	0.86	0.78
		1	0.82	0.66	0.8	0.71	0.82
		2	0.82	0.67	0.89	0.76	0.78
	WizardLM	0	0.8	0.71	0.86	0.82	0.65
		1	0.8	0.7	0.89	0.83	0.73
		2	0.79	0.7	0.89	0.78	0.67
AUC	Llama 2-Chat	0	0.65	0.64	0.88	0.80	0.65
		1	0.51	0.59	0.65	0.79	0.63
		2	0.49	0.54	0.65	0.81	0.47
	Vicuna	0	0.73	0.83	0.997	0.996	0.71
		1	0.67	0.74	0.96	0.97	0.63
		2	0.54	0.74	0.91	0.95	0.50
	WizardLM	0	0.64	0.81	0.83	0.97	0.59
		1	0.64	0.79	0.95	0.96	0.71
		2	0.57	0.79	0.92	0.96	0.62

when using the same bLLM, the optimal prompt template can vary depending on the dataset under consideration.

Specifically, regarding the macro-F1 score, *Prompt 0* emerges as the most effective choice, yielding the highest scores in 10 instances. Following closely, *Prompt 1* leads in 6 instances. Interestingly, *Prompt 2*, while achieving the top performance on only 4 occasions, occasionally surpasses *Prompt 1* by a significant margin, notably in the case of the Llama 2-Chat within the StackOverflow dataset. Regarding the micro-F1 scores, both *Prompt 0* and *Prompt 1* achieved the highest scores 7 times, while *Prompt 2* ranked the first 5 times. Regarding the AUC value, *Prompt 0* emerges as the most effective choice, yielding the highest scores in 12 instances. *Prompt 1* and *Prompt 2* achieves the top performance on only 2 and 1 occasions, respectively. These results show that Prompt 0 can achieve overall best results considering all the metrics.

Table 3 provides an insight into the variance within each result group under the zero-shot setting. We define a result *group* as results generated by the same bLLM when applied to the same dataset with varying prompts. This analysis aims to underscore the impact of prompt selection on

Table 3. Zero-shot: Score difference between the highest and lowest ones with each LLM on one dataset and the value in the parenthesis shows the difference percentage.

	Model	Gerrit	GitHub	GooglePlay	Jira	StackOverflow
Macro-F1	LLAMA 2-CHAT	0.04 (5.6%)	0.04 (6.3%)	0	0.12 (16.9%)	0.06 (13.3%)
	VICUNA	0.03 (4.3%)	0.07 (10.8%)	0.24 (32.4%)	0.16 (23.2%)	0.06 (11.3%)
	WIZARDLM	0.01 (1.5%)	0.01 (1.4%)	0.03 (3.8%)	0.05 (6.5%)	0.18 (43.9%)
	Avg. Diff	3.80%	6.17%	12.07%	15.53%	22.83%
Micro-F1	LLAMA 2-CHAT	0.01 (1.2%)	0.04 (6.3%)	0.03 (3.3%)	0.13 (18.3%)	0.11 (18.0%)
	VICUNA	0.01 (2.1%)	0.06 (12.5%)	0.17 (35.4%)	0.15 (31.3%)	0.04 (8.3%)
	WIZARDLM	0.01 (1.3%)	0.01 (1.4%)	0.03 (3.5%)	0.05 (6.4%)	0.08 (12.3%)
	Avg. Diff	1.5%	6.7%	14.0%	18.7%	12.9%
AUC	LLAMA 2-CHAT	0.16 (31.9%)	0.09 (17.3%)	0.23 (36.0%)	0.02 (2.3%)	0.18 (37.7%)
	VICUNA	0.19 (34.6%)	0.09 (11.8%)	0.09 (9.4%)	0.05 (5.4%)	0.21 (42.1%)
	WIZARDLM	0.07 (12.8%)	0.02 (2.8%)	0.13 (15.2%)	0.02 (1.7%)	0.12 (20.2%)
	Avg. Diff	26.4%	10.6%	20.2%	3.1%	33.4%

performance. Remarkably, within the same group, we observe disparities as substantial as 43.9%. Expanding our examination to encompass different models operating on identical datasets reveals an average difference as substantial as 22.83%. This discovery underscores the sensitivity of bLLMs to the choice of prompts in zero-shot learning.

Based on the results of the Wilcoxon signed-rank test with Bonferroni correction for multiple comparisons, we can draw the following conclusions regarding the significance of differences between the distinct prompt templates:

- The comparison between prompt 0 and prompt 1 yielded a p-value of 0.0520, which is greater than the corrected alpha threshold of 0.0167. Therefore, we fail to reject the null hypothesis, indicating no significant difference between prompt 0 and prompt 1.
- The comparison between prompt 0 and prompt 2 resulted in a p-value of 0.0048, which is less than the corrected alpha threshold of 0.0167. Consequently, we reject the null hypothesis, indicating a significant difference between prompt 0 and prompt 2.
- The comparison between prompt 1 and prompt 2 produced a p-value of 1.0, which is much greater than the corrected alpha threshold of 0.0167. Thus, we fail to reject the null hypothesis, indicating no significant difference between prompt 1 and prompt 2.

In summary, the analysis reveals a significant difference between prompt 0 and prompt 2, while no significant differences were found between the other pairs of zero-shot prompt templates.

Although the overall difference between different prompt templates on the same model is not significant, the best performance difference between different prompts on the same model can be as large as 43.9%. Therefore, it is crucial to test multiple prompt templates and select the one that yields the highest accuracy for a given task and dataset. Based on the experimental results, we have the following implications: a) Researchers should test a range of prompts when evaluating models for the SA4SE task, rather than relying on generic prompts. b) Practitioners implementing SA tools for SE domain (e.g., analyzing code reviews or developer forums) should invest time in prompt optimization for their specific use case. c) The variability in results suggests that ensemble

Table 4. Few-shot Performance: Comparative Results of LLMs Across Five Datasets. Cells highlighted in red indicate the highest scores achieved among the three prompts executed by each respective model.

	Model	Shot	Gerrit	GitHub	GooglePlay	Jira	StackOverflow
Macro-F1	Llama 2-Chat	1	0.69	0.54	0.89	0.82	0.42
		3	0.69	0.60	0.87	0.84	0.46
		5	0.68	0.61	1	0.89	0.47
	Vicuna	1	0.74	0.68	0.74	0.77	0.56
		3	0.73	0.72	0.82	0.86	0.65
		5	0.71	0.72	0.77	0.89	0.64
	WizardLM	1	0.76	0.68	0.89	0.78	0.54
		3	0.75	0.72	0.87	0.9	0.59
		5	0.75	0.71	0.82	0.91	0.54
Micro-F1	Llama 2-Chat	1	0.78	0.54	0.94	0.83	0.42
		3	0.79	0.60	0.94	0.86	0.51
		5	0.77	0.60	1	0.89	0.59
	Vicuna	1	0.82	0.69	0.8	0.78	0.78
		3	0.81	0.72	0.89	0.87	0.83
		5	0.78	0.72	0.83	0.89	0.82
	WizardLM	1	0.83	0.67	0.94	0.79	0.67
		3	0.84	0.72	0.94	0.91	0.74
		5	0.83	0.70	0.91	0.92	0.74
AUC	Llama 2-Chat	1	0.62	0.63	0.72	0.85	0.63
		3	0.52	0.62	0.77	0.80	0.58
		5	0.57	0.57	0.76	0.83	0.58
	Vicuna	1	0.71	0.75	0.96	0.92	0.65
		3	0.67	0.74	0.95	0.99	0.65
		5	0.73	0.75	0.96	0.98	0.65
	WizardLM	1	0.74	0.72	0.92	0.92	0.56
		3	0.63	0.72	0.93	0.99	0.58
		5	0.59	0.71	0.87	0.98	0.53

methods, combining multiple prompts or models, might offer more robust performance in real-world applications. Our findings also open up an avenue for future research: Developing systematic methods for prompt optimization in SE domain, including but not limited to the SA4SE task.

> **Answer to RQ1:** In the SA4SE context, it is evident that bLLMs exhibit sensitivity to prompts in zero-shot learning scenarios. When employing various prompt templates, the average macro-F1 score difference spans from 3.8% to 22.83%, the average micro-F1 score difference ranges from 1.5% to 18.7%, and the average AUC value difference varies from 3.1% to 33.4%.

4.2 RQ2: Impact of different shots on the performance of bLLMs with few-shot learning

Table 4 showcases the outcomes of few-shot learning utilizing three distinct bLLMs across five diverse datasets. It is important to reiterate that a result "group" signifies the results produced by the same bLLM when applied to the same dataset with varying shot numbers.
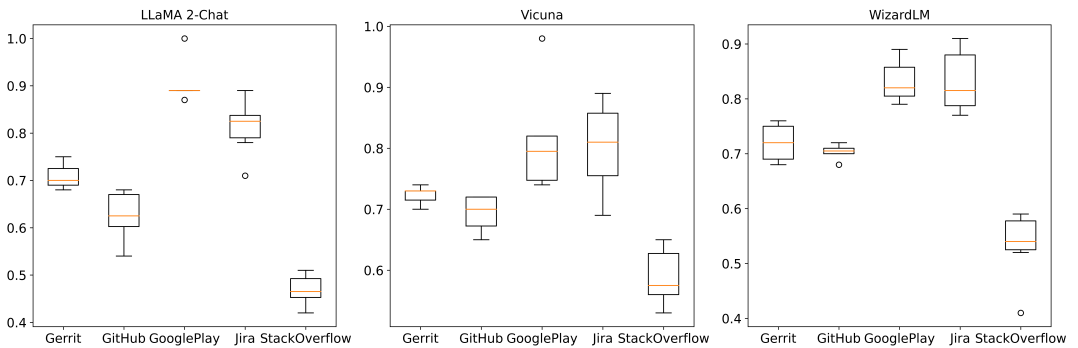
Fig. 4. Sensitivity of different prompt designs. The circles depicted in the figure represent outlier data points.

In summary, when considering macro-F1, the 5-shot configuration emerges as the leader in 7 instances, followed by the 3-shot setup in 5 cases, and the 1-shot configuration excels in 4 instances. Regarding micro-F1, the 5-shot configuration is at the top 7 times, the 3-shot setup prevails 9 times, and the 1-shot configuration leads twice. Generally, the trend for both macro-F1 and micro-F1 indicates that having more than one example is more beneficial than having only one.

Among the three models, LLAMA 2-CHAT consistently benefits from having more examples, except for the `Gerrit` dataset concerning macro-F1 and micro-F1. In the case of VICUNA and WIZARDLM, the impact of additional examples is noticeable primarily on the `Jira` dataset, affecting both macro-F1 and micro-F1. This underscores the fact that the influence of additional examples can vary depending on the bLLM and dataset employed. This can be explained by recognizing that SA, especially the sentiment classification task central to our work, is relatively straightforward. Consequently, the effects of increasing training examples might not be as pronounced. All bLLMs show improved performance with more shots on the `Jira` dataset, likely due to its binary classification nature. Despite being prompted to output only [*"negative"*, *"positive"*] labels, bLLMs tend to predict *neutral* sentiments. This tendency decreases as the number of shots in the prompt increases. With only two labels in the `Jira` dataset, fewer "neutral" predictions lead to higher macro- and micro-F1 scores. This suggests that additional shots significantly reduce bLLMs' misclassification tendencies, especially in datasets with limited label options like `Jira`.

We take VICUNA's performance as an example to illustrate this trend. In zero-shot settings, it explicitly predicts "neutral" 19 times on the `Jira` dataset with the Prompt 1. This issue diminishes in few-shot settings: VICUNA predicts "neutral" 11, 8, and 5 times with 1-shot, 3-shot, and 5-shot prompt templates, respectively.

Moving on from the advantages of more shots, we also observed a decline in macro-F1 scores with an increase in the number of examples. This trend is particularly noticeable when applying all three bLLMs to the `Gerrit` dataset and when using WIZARDLM on the `GooglePlay` dataset. One plausible explanation for this phenomenon is that an increased number of examples leads to longer prompts, which could potentially confuse the bLLMs. As indicated by Table 1, the documents from the `Gerrit` dataset have the highest average number of tokens. Consequently, introducing more examples results in even lengthier prompts. This observation aligns with prior research on bLLMs in the broader SA domain [72].

Figure 4 provides a clearer illustration of this phenomenon. The box plot delves into the variance of macro-F1 scores achieved by different prompts for each model on each dataset. We have examined our study's six prompt templates. This figure reveals that the influence of different prompts on
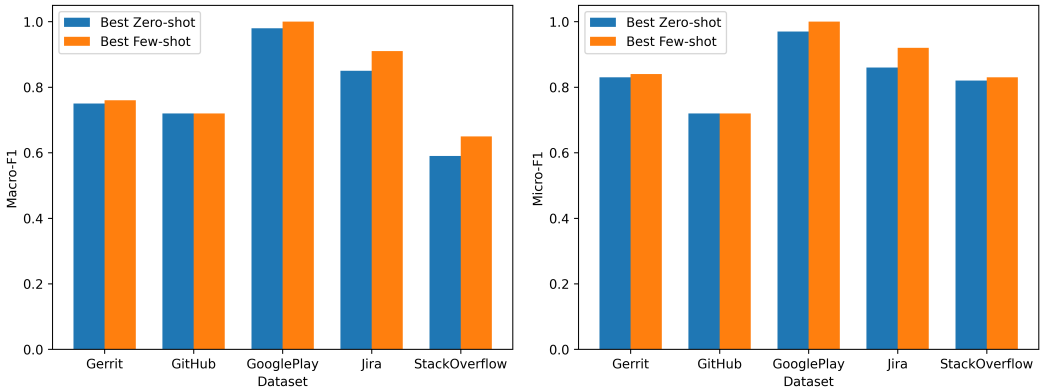
Fig. 5. Comparison of the highest macro-F1 and micro-F1 scores achieved through zero-shot learning and few-shot learning.

performance varies depending on both the model and the dataset. In general, the models demonstrate differing levels of sensitivity to prompts. Notably, on the `Jira` dataset, all models exhibit high sensitivity to prompts, signifying that the choice of prompt has a substantial impact on results. In contrast, on the `Gerrit` and `GitHub` datasets, models appear less responsive to different prompts, suggesting that the choice of prompt has a relatively smaller effect on their performance.

We also compare the best results of any of the three bLLMs under few-shot learning and those achieved under zero-shot learning. Figure 5 illustrates that, in all the five datasets, the highest macro-F1 score achieved through few-shot learning either surpasses or equals the highest macro-F1 score attained via zero-shot learning. We can observe a similar trend in terms of the micro-F1 scores. This trend is particularly pronounced on the `Jira` dataset, where bLLMs perform notably better under the few-shot learning paradigm. Furthermore, to assess whether the results of few-shot learning are significantly better than zero-shot learning, we conducted a Wilcoxon signed-rank test on one pairwise comparison: the zero-shot results from *Prompt 0* versus the five-shot results. Notably, this comparison yielded a p-value of 0.674. This leads us to retain the null hypothesis (H0.2), indicating that the observed differences in results did not reach statistical significance. Hence, it is important to recognize that while few-shot learning does demonstrate superior results, the margin of improvement is not substantial.

> **Answer to RQ2:** Although the top-performing bLLM achieved equal or higher macro- and micro-F1 scores in all five datasets with few-shot learning, the performance gap between few-shot and zero-shot learning was insignificant. In addition, there is no guarantee that the same bLLM will exhibit improved performance through few-shot learning over zero-shot learning.

4.3 RQ3: Comparison between fine-tuned sLLMs and bLLMs

Table 5 presents a comparative analysis of the top-performing results obtained through two distinct approaches: prompting bLLMs and fine-tuning sLLMs.

On the `GooglePlay` dataset, where the fine-tuning data is limited to fewer than 300 data points and with a negative:neutral:positive ratio of 26:5:37, bLLMs demonstrate a remarkable performance advantage. The most effective bLLM, LLAMA 2-CHAT, significantly enhances the performance of the leading sLLM, DistilBERT, by a substantial improvement of 75.4%. It indicates that when labeled

Table 5. Results of LLMs compared with fine-tuned sLLMs. Cells highlighted in red indicate the highest scores achieved among the three prompts executed by each respective model.

		Model	Gerrit	GitHub	GooglePlay	Jira	StackOverflow
Macro-F1	bLLM	LLAMA 2-CHAT	0.75	0.68	1	0.89	0.51
		VICUNA	0.74	0.72	0.98	0.89	0.65
		WIZARDLM	0.76	0.72	0.89	0.91	0.59
	sLLM	ALBERT	0.73	0.90	0.56	0.97	0.64
		BERT	0.76	0.92	0.49	0.95	0.50
		DistilBERT	0.81	0.92	0.57	0.95	0.60
		RoBERTa	0.74	0.94	0.42	0.95	0.68
		XLNet	0.77	0.91	0.39	0.94	0.67
Micro-F1	bLLM	LLAMA 2-CHAT	0.83	0.68	1	0.89	0.72
		VICUNA	0.82	0.72	0.97	0.89	0.83
		WIZARDLM	0.84	0.72	0.94	0.92	0.74
	sLLM	ALBERT	0.81	0.90	0.80	0.97	0.84
		BERT	0.80	0.92	0.71	0.96	0.84
		DistilBERT	0.86	0.92	0.83	0.96	0.84
		RoBERTa	0.81	0.94	0.63	0.95	0.86
		XLNet	0.83	0.91	0.57	0.95	0.89
AUC	bLLM	LLAMA 2-CHAT	0.65	0.64	0.88	0.85	0.65
		VICUNA	0.73	0.83	0.997	0.996	0.71
		WIZARDLM	0.74	0.81	0.95	0.99	0.71
	sLLM	ALBERT	0.83	0.98	0.67	0.998	0.87
		BERT	0.89	0.98	0.80	0.998	0.81
		DistilBERT	0.89	0.99	0.89	0.996	0.82
		RoBERTa	0.87	0.99	0.69	0.99	0.87
		XLNet	0.89	0.98	0.73	0.99	0.86

data is scarce and the training dataset is highly imbalanced, fine-tuning bLLMs should be the preferred approach over sLLMs.

On the contrary, we have observed that fine-tuning sLLMs produces superior results on the GitHub and Jira datasets. The GitHub dataset benefits from a larger training dataset and a more evenly distributed class structure, making it particularly well-suited for fine-tuning sLLMs. RoBERTa outperforms the best-performing bLLMs VICUNA and WIZARDLM by 30.6%. In contrast, the Jira dataset, though smaller than GitHub, offers a more favorable class distribution than GooglePlay, with a negative-to-positive ratio of approximately 1:2. The best-performing sLLM, ALBERT, outperforms the best-performing bLLM, WIZARDLM by 6.6%.

For the Gerrit and StackOverflow dataset, both bLLMs and sLLMs exhibit relatively modest performance. The best-performing bLLM achieves a macro-F1 score of 0.65, while the top sLLM achieves a macro-F1 score of 0.68 on the StackOverflow dataset. The corresponding score is 0.76 and 0.81 on the Gerrit dataset. Both datasets share the challenge of imbalanced label distribution, which explains the limited success of sLLMs. Furthermore, for the StackOverflow dataset, given the short length of the sentences in this dataset, neither bLLMs nor sLLMs may have sufficient contextual information to make accurate sentiment predictions.
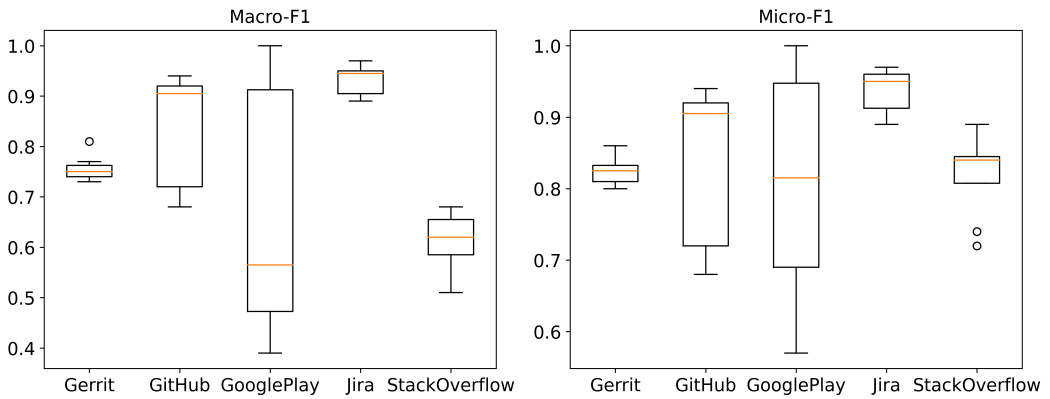
Fig. 6. Performance variance of all the models on each dataset.

Moreover, in Figure 6, we observe the performance of all the models across different datasets. Notably, on the `GooglePlay` dataset, there is a significant variance, with bLLMs standing out by achieving the highest macro-F1 and micro-F1 scores. This underscores bLLMs' superiority over sLLMs on this specific dataset.

On the other hand, for the `Gerrit`, `Jira`, and `StackOverflow` datasets, the variance is comparatively smaller, suggesting that while sLLMs outperform bLLMs, the margin of difference is not substantial.

Conversely, in the case of the `GitHub` dataset, sLLMs demonstrate a substantial advantage over bLLMs. This discrepancy is likely due to the abundance of training data available for the `GitHub` dataset. It reinforces that sLLMs are most effective when ample, balanced training data is available.

To evaluate the significance of differences between bLLMs and sLLMs, we performed a Wilcoxon signed-rank test on one pair of comparisons: the overall best bLLM, i.e., LLAMA 2-CHAT, versus the overall best sLLM, i.e., RoBERTa. The comparison resulted in a p-value of 0.277, leading us to retain the null hypothesis (H0.3). This indicates that the observed differences in results did not achieve statistical significance.

> **Answer to RQ3:** In scenarios with limited labeled data and pronounced class imbalance, prompting bLLMs is a more effective strategy, outperforming fine-tuning sLLMs, e.g., in the `GooglePlay` dataset, LLAMA 2-CHAT outperforms the DistilBERT by 75.4%. In contrast, when ample training data is available and the dataset demonstrates a balanced distribution, the preference should lean toward sLLMs as the more suitable approach, e.g., in the `GitHub` dataset, RoBERTa outperforms VICUNA by 30.6%.

4.4 RQ4: Error Analysis

Quantitative Analysis. In Figure 7, it is evident that all three bLLMs consistently generate accurate predictions in a substantial portion of instances. Specifically, in the realm of zero-shot learning, these models collectively predicted the correct sentiments in 73.7% cases, and in the context of few-shot learning, they made the correct prediction in 61.4% cases. Notably, both VICUNA and WIZARDLM stand out as the bLLMs with the highest degree of overlapping predictions across both scenarios. In the zero-shot context, they share common predictions in an impressive 83.3% of their respective correct predictions, while in the few-shot scenario, this shared correct prediction rate remains substantial at 70.1%. Conversely, the lowest degree of common correct predictions

(a) Zero-shot learning with Prompt 0   (b) Few-shot learning with 5-shot prompt
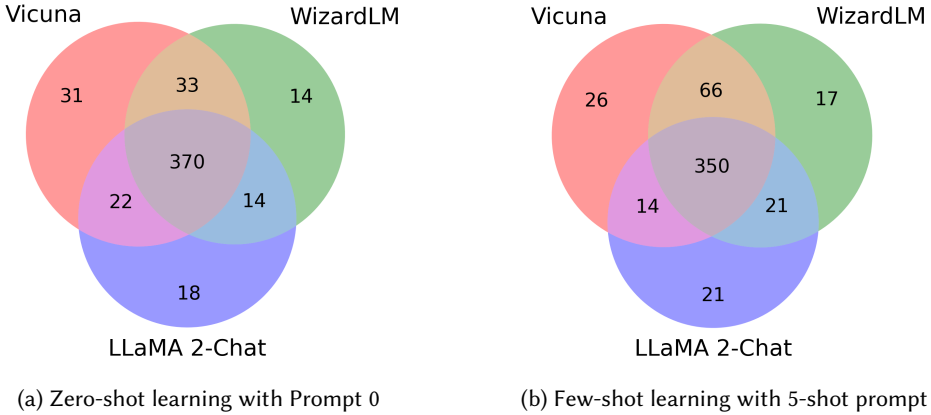
Fig. 7. The Venn diagram of the correct predictions made by bLLMs.

Table 6. Overlap in Misclassification Across LLMs in Zero-Shot and Few-Shot Settings. The *Common* Column Indicates Misclassifications Shared Between Both Settings.

	Misclassified (% of the test set)			Test set size
	Zero-shot	*Few-shot*	*Common*	
Gerrit	15 (13.2%)	10 (8.8%)	8 (7.0%)	114
GitHub	39 (15.6%)	41 (16.4%)	23 (9.2%)	250
GooglePlay	1 (2.9%)	0	0	35
Jira	8 (10.5%)	4 (5.3%)	4 (5.3%)	76
StackOverflow	19 (17.4%)	14 (12.8%)	11 (10.1%)	109
Total	82 (14%)	69 (11.8%)	46 (7.9%)	584

is observed between Vicuna and Llama 2-Chat, with rates of 80.3% and 69.1% for zero-shot and few-shot scenarios, respectively. These results underscore the bLLMs' capacity to achieve comparable success rates in most cases while also emphasizing their unique strengths.

Now, we shift our focus to the errors made by these bLLMs. Table 6 demonstrates again that overall, few-shot learning is more effective than zero-shot learning, as all the bLLMs misclassified more cases under zero-shot learning. However, we also notice that the number of common misclassification by both settings is 46, which accounts for 7.9% of the total test cases. To better understand the difficulties and challenges faced by bLLMs on the task of SA4SE, we manually examined these cases. In our experiments, we analyze the cases where all the bLLMs did not get any correct prediction no matter the setting.

Qualitative Analysis. Table 7 shows the error categories of these common misclassifications by all the bLLMs. *Polar facts* emerge as the most prominent category, representing the majority of failure cases. In some cases, the sentence describes a fact, which may usually invoke for most people a positive or negative feeling, i.e. the annotator considered the described situation either as desirable or undesirable. They have been annotated inconsistently across different datasets. For

Table 7. Distribution of error categories and their percentage among the whole error cases.

Error Category	# Cases (%)
Polar facts	16 (34.8%)
Subjectivity in annotation	13 (28.3%)
General error	8 (17.4%)
Politeness	5 (10.9%)
Implicit sentiment polarity	4 (8.7%)

example, in `GitHub` and `StackOverflow` datasets, they were labeled as *neutral*. However, in `Gerrit` and `Jira` datasets, they were labeled as *negative*. For instance, in the `Jira` dataset, *There is no need to reference $wizard, it's an object* was labeled as negative. In the `GitHub` dataset, *Ok, I'll fix them.* was labeled as neutral.

The second most prevalent category, *Subjectivity in annotation*, comprises cases where the evaluators' interpretation of sentiment differed from the originally assigned label. As recognized in prior works [33], sentiment or option itself is subjective. Similarly, sentiment annotation is also a subjective activity. Depending on personality traits or personal disposition, different annotators' perceptions of emotions might vary [52]. Thus, it is not rare that the evaluators have different perceptions from the original annotators. One instance is from the `StackOverflow` dataset, *Less likely to be blocked by paranoid firewall configurations.*. The evaluators consider this sentence as *positive*, while the ground-truth label is *negative*. Depending on which perspective we think, both make sense: we consider it positive, as we focus on "less likely to be blocked." However, the annotators may give more weight to "paranoid firewall configurations".

General errors account for 17.4% of cases and occur when the model fails to identify clues in the document that would be readily apparent to a human. For instance, emoticons can signal sentiment, as observed in the sentence from the `GitHub` dataset, "yep, it's work, but I need to add user and password for proxy connection=('." This sentence may convey negative sentiment to a human, particularly due to the emoticon "=(" embedded within it.

Politeness contributes to 10.9% of the error cases, arising when the presence of phrases like "thanks" or "sorry" leads to inconsistencies across different datasets. For instance, in the `GitHub` dataset, the sentence "sorry I did not realize you were already there..." was labeled as "negative", although some individuals may perceive it as "neutral". Similarly, in the `StackOverflow` dataset, "Good luck!" was labeled as "neutral", but certain interpretations could classify it as "positive". These inconsistencies pose challenges for models when predicting labels with limited examples.

Lastly, *Implicit sentiment polarity* accounts for 8.7% error cases. When there is a lack of explicit sentiment clues, it could be hard to decide which sentiment is contained. For instance, *Yes, it did not cause message loss just unnecessary retransmits.*, this sentence was annotated as negative. However, there is no obvious sentiment clue.

In summary, due to the inconsistency in labeling rules and the subjective nature of the task, challenges arise where bLLMs may struggle to improve significantly. However, in the case of general errors, there is a potential for improvement as bLLMs continue to advance.

> **Answer to RQ4:** In 7.9% of test cases, all the bLLMs consistently fail to make correct predictions, regardless of the setting. The inconsistency in labeling rules and the inherently subjective nature of SA4SE remain challenges for bLLMs.

5 DISCUSSION

5.1 Additional Experiments

In addition to our primary experiments, we carried out one supplementary set of experiments to further enrich and substantiate our study.

An Alternative Way to Leverage sLLMs. Building on existing research [71], we adapt sLLMs for sentiment label prediction. An alternative approach involves leveraging sLLMs' embeddings in conjunction with various machine learning (ML) classifiers. Our study investigates the efficacy of this methodology. Specifically, we evaluate five sLLMs, as discussed in Section 3.4, in conjunction with three distinct ML classifiers: Naïve Bayes [50], Decision Tree [48], and Random Forest [5]. These ML models are implemented using the scikit-learn library [46], utilizing their default settings. We take the first token ([CLS]) embedding for each document in our dataset as document representation.

The results presented in Table 8 offer a comprehensive comparison between the efficacy of fine-tuning sLLMs and employing ML models with static sLLM embeddings. It is observed that, in most cases, fine-tuning sLLMs leads to superior macro-F1 scores compared to traditional ML classifiers with fixed embeddings. A notable exception to this trend is observed with the Naïve Bayes classifier on the GooglePlay dataset. In our evaluation of ML classifiers, Random Forest stands out, achieving the highest average macro-F1 score of 0.57, micro-F1 score of 0.77, and AUC value of 0.79 across various datasets and embedding models. In comparison, the approach of fine-tuning sLLMs results in an average macro-F1 score of 0.75, which is a significant 32.3% improvement over the Random Forest model. Additionally, fine-tuning sLLMs achieves an average micro-F1 score of 0.85, surpassing the Random Forest model by 10.5%. Fine-tuning sLLMs can achieve an average AUC value of 0.89, which is 12.4% higher than the Random Forest model. This demonstrates the considerable advantages of fine-tuning sLLMs in terms of all the macro-F1, micro-F1, and AUC scores.

5.2 Implications for Future Research

Based on the experimental results in our study, we derive the empirical guidelines for future research on SA4SE and SE in general as follows.

Select your approach: considering the size and class distribution of the dataset. When determining the most suitable strategy for a specific task, it is important to consider the size and class distribution of the dataset. Based on our empirical results, it is crucial to recognize that, in cases with ample and balanced training data, fine-tuning sLLMs remains the preferred choice. This guideline is applicable to numerous SE tasks. If there are already manually curated datasets available or acquiring labeled data is not a significant challenge, fine-tuning sLLMs represents a straightforward and effective option. However, in scenarios where labeled datasets are scarce, bLLMs emerge as a potential solution.

Effective prompt engineering unlocks the full potential of bLLMs. Our experiments reveal a crucial insight: while prompt templates may appear similar at first glance, determining which one will yield the highest accuracy requires actual execution and template refinement. Carefully crafting prompt templates proves advantageous, particularly in zero-shot scenarios, as it allows us to better leverage the capabilities of bLLMs. For future research, we recommend testing multiple prompt templates on a subset of samples and selecting the best-performing ones for deployment. This meticulous process of prompt engineering and evaluation is essential for optimizing the

Table 8. Comparative results of best fine-tuned sLLMs and traditional machine learning models using sLLMs embeddings. Cells highlighted in red indicate the highest scores achieved among different models.

			Gerrit	GitHub	Google Play	Jira	StackOverflow
Macro-F1	**Naive Bayes**	ALBERT	0.53	0.64	0.42	0.84	0.43
		BERT	0.52	0.65	0.58	0.88	0.53
		DistilBERT	0.51	0.64	0.56	0.81	0.44
		RoBERTa	0.46	0.6	0.29	0.62	0.41
		XLNet	0.52	0.41	0.46	0.74	0.29
	Decision Tree	ALBERT	0.63	0.56	0.42	0.89	0.48
		BERT	0.5	0.61	0.54	0.82	0.37
		DistilBERT	0.52	0.56	0.48	0.88	0.33
		RoBERTa	0.57	0.55	0.48	0.75	0.31
		XLNet	0.58	0.44	0.25	0.68	0.4
	Random Forest	ALBERT	0.49	0.73	0.37	0.92	0.3
		BERT	0.46	0.73	0.52	0.97	0.3
		DistilBERT	0.53	0.76	0.51	0.92	0.39
		RoBERTa	0.43	0.74	0.48	0.94	0.3
		XLNet	0.46	0.54	0.3	0.78	0.3
	Fine-tuning	Best	0.81	0.94	0.57	0.97	0.68
Micro-F1	**Naive Bayes**	ALBERT	0.55	0.64	0.51	0.86	0.57
		BERT	0.56	0.66	0.69	0.89	0.65
		DistilBERT	0.54	0.65	0.66	0.83	0.54
		RoBERTa	0.47	0.63	0.43	0.64	0.49
		XLNet	0.54	0.48	0.51	0.78	0.49
	Decision Tree	ALBERT	0.72	0.57	0.57	0.91	0.71
		BERT	0.65	0.62	0.6	0.84	0.68
		DistilBERT	0.62	0.56	0.69	0.89	0.66
		RoBERTa	0.68	0.56	0.66	0.79	0.64
		XLNet	0.68	0.46	0.34	0.71	0.7
	Random Forest	ALBERT	0.75	0.74	0.57	0.93	0.8
		BERT	0.75	0.74	0.74	0.97	0.8
		DistilBERT	0.77	0.77	0.74	0.93	0.82
		RoBERTa	0.75	0.76	0.71	0.95	0.8
		XLNet	0.75	0.59	0.49	0.84	0.8
	Fine-tuning	Best	0.86	0.94	0.83	0.97	0.89
AUC	**Naive Bayes**	ALBERT	0.62	0.79	0.69	0.91	0.64
		BERT	0.57	0.83	0.73	0.93	0.81
		DistilBERT	0.57	0.79	0.76	0.91	0.74
		RoBERTa	0.59	0.79	0.6	0.75	0.69
		XLNet	0.62	0.65	0.68	0.8	0.56
	Decision Tree	ALBERT	0.63	0.67	0.59	0.89	0.63
		BERT	0.5	0.71	0.64	0.84	0.53
		DistilBERT	0.52	0.68	0.64	0.88	0.5
		RoBERTa	0.57	0.67	0.65	0.75	0.48
		XLNet	0.58	0.58	0.43	0.69	0.56
	Random Forest	ALBERT	0.73	0.89	0.76	0.99	0.72
		BERT	0.68	0.91	0.77	0.996	0.69
		DistilBERT	0.65	0.9	0.84	0.99	0.68
		RoBERTa	0.64	0.9	0.79	0.98	0.72
		XLNet	0.67	0.76	0.67	0.85	0.63
	Fine-tuning	Best	0.89	0.99	0.89	0.998	0.87

performance of bLLMs. Furthermore, our findings suggest that in few-shot learning scenarios, adding more examples (shots) is not always beneficial. Depending on the task characteristics, a longer context can potentially confuse bLLMs and lead to suboptimal results. Therefore, researchers should carefully evaluate the impact of increasing the number of shots, as it may not consistently improve performance.

Comparison of prompt engineering for LLMs in the SA4SE task and other SE tasks. The SA4SE task is relatively straightforward, with the primary goal of determining the sentiment of a given text from SE artifacts. This simplicity allows for the creation of concise and effective prompt templates. Thus, this study focuses on investigating whether minor prompt changes would lead to significant performance differences in the models. However, some other SE tasks, such as code summarization [2], bug replay [20], and vulnerability detection [75], often contain additional contextual information that can be included in the prompt. For instance, the prompt used for code summarization [2] includes repository information, variable names, and scopes. Such information is unavailable in the SA4SE task, where the prompt is limited to the text to be analyzed. These task-specific details can guide the model in generating the desired output. In these SE tasks, prompt engineering may focus on which pieces of information to include in the prompt to guide the model's output generation. In contrast, for SA4SE, prompt engineering may focus more on word choice and whether to include additional examples in the prompt. Given the diversity of SE tasks, future research should explore developing task-specific prompt templates to maximize the performance of LLMs. This tailored approach can enhance the adaptability of LLMs to various SE tasks, ensuring optimal performance across different domains.

Additional guidelines for rule setting in human-labeled SA datasets or prompt design. As elaborated in Section 4.4, the inconsistency in labeling practices across various datasets poses a significant challenge for bLLMs to predict labels accurately. To enhance performance, we propose two potential approaches: 1. Encouraging human annotators to adhere to general labeling rules when annotating data. 2. Empowering bLLMs to incorporate dataset-specific labeling rules. Recall that despite restricting the label options to "positive" and "negative", bLLMs still occasionally return "neutral" labels when assessing the Jira dataset. This observation underscores the importance of further exploring and refining prompts to match the dataset characteristics.

5.3   Threats to Validity

Threats to Internal Validity. One potential source of bias in our empirical study may arise from the choice of prompt templates. To address this concern, we conducted experiments in the zero-shot setting using three different prompt templates. It is important to note that we based our prompt templates on prior work [72]. Additionally, we examined the influence of the number of shots in the few-shot setting. Furthermore, there is a potential concern regarding the quality of the labeled dataset. We did not generate new datasets but rather relied on pre-existing ones from other sources. Consequently, we inherit this quality concern from the original works. However, as described in Section 3.3, in the original labeling process, each dataset was labeled by two or more labelers labeled individually and resolved the conflict by involving another labeler. Thus, we consider the threat to be minimal.

Threats to External Validity. Our findings may not necessarily generalize to data from other platforms. Nevertheless, we have taken steps to mitigate this threat by considering data from five distinct platforms. It is important to recognize that our results are specific to the dataset and experimental setup we employed. In the few-shot learning setting, our results are contingent on randomly sampled examples. Nevertheless, our experiments and results still offer valuable insights,

demonstrating that bLLMs can be a promising approach when dealing with a scarcity of annotated data. In the future, we plan to expand our analysis by incorporating additional datasets from various platforms and exploring more diverse prompt templates to enhance our understanding of leveraging bLLMs for SE in SE further.

Another potential threat is our reliance on a single held-out test set for evaluation. While this approach is common in the SA4SE literature [7, 41, 43, 71], it may not fully capture the diversity of scenarios and data distributions that the models are intended to handle. Many other SE tasks utilizing bLLMs also adopt the held-out test set approach, such as code comment generation [21], log parsing [37], and automatic logging [65]. Beyond the SE research areas, other domains, including natural language processing and information retrieval, have also employed the held-out setting for evaluation, e.g., stream recommendation [68], document-level sentiment analysis [55], generative outfit recommendation [66]. Although cross-validation may provide a more comprehensive assessment of performance across different situations, we believe the held-out test set approach is sufficient for our study, given its wide adoption in the literature. In the future, we plan to incorporate cross-validation into our evaluation methodology to enhance the robustness and reliability of our findings.

The additional potential threat to external validity is the chosen bLLMs can be surpassed by newer models. Despite this, our research is still relevant as it offers crucial insights into the current capabilities and constraints of the accessible open-source bLLMs. It establishes a benchmark for evaluating the progression of bLLMs over time. By documenting the specific versions of the models utilized, we guarantee the reproducibility of our findings. This commitment ensures that our work serves as a foundational reference, facilitating subsequent research and exploration in the evolving domain of bLLMs on the SA4SE task.

Threats to Construct Validity. The first consideration for the construct validity of our study is the potential for data leakage, particularly given our use of sLLMs and bLLMs. Data leakage could occur if the evaluation dataset was already present in the pre-training dataset of the models, potentially inflating model performance. However, we assess this risk to be low in our study. The dataset we employed is not readily accessible through standard web browsing and requires downloading from a specific URL. This aspect reduces the likelihood that it was included in the LLM's pre-training data, mitigating the risk of data leakage affecting our results.

Another potential threat is the unreliable labeled data. As discussed in Section 4.4, a significant challenge bLLMs encountered was the inconsistency in labeling rules and guidelines within the ground truth dataset. This inconsistency poses a threat to the construct validity of our study, as it can lead to confusion in model training, potentially resulting in suboptimal performance. To address this issue and enhance the reliability of our findings, future research should focus on establishing clear, consistent labeling rules prior to the creation of new datasets. Additionally, a thorough manual review of existing labeled datasets is recommended to ensure uniformity and accuracy, thereby mitigating the risks associated with inconsistent labeling practices.

In addition, our choice to label "neutral" as the opposite of the ground truth in the Jira dataset introduces an additional risk. This approach was adopted to establish a lower-bound for performance, which might, however, compromise the effectiveness of bLLMs. Similarly, we also map non-explicit responses from bLLMs as "neutral" poses a potential threat. This decision, aimed at ensuring consistency in SA, is based on the rationale that "mixed" sentiments typically indicate a balanced or uncertain position, closely aligning with a *neutral* stance.

Lastly, to obtain prediction probability scores from bLLMs, we directly prompt the models for their probability scores for each label. In cases where the total sum of these scores does not equal 1, we normalize them by dividing each score by the total sum. However, this approach does not extract

the probability scores directly from the model's internal computations, which may not be the most optimal method for obtaining accurate probability scores. This could potentially introduce noise into our results, as the probability scores may not align with the actual probabilities. For instance, in the `GooglePlay` dataset, the LLAMA 2-CHAT model made perfect predictions on the test set with 5-shot learning, but the AUC value was only 0.76. In one case, although the correct prediction label was negative, the predicted probability for the negative label was only 0.2, while the probability for the positive label was 0.8. While such cases are rare, we believe this threat is minimal. Moreover, our main evaluation metric is macro-F1, which is calculated based on the precision and recall of each class and does not consider the probability scores of the models. Thus, our study is not highly affected by potential inaccuracies in the AUC metric. To mitigate this threat, we plan to explore alternative methods for extracting probability scores from bLLMs in future research.

6 CONCLUSION AND FUTURE WORK

In conclusion, our study marks the initial step towards comprehending the potential of utilizing prompting bLLMs in discerning sentiment within SE domain documents. Our experiments reveal that in cases with limited annotated data, bLLMs outperform sLLMs, and zero-shot learning is a viable approach. However, when substantial and well-balanced training data is available, fine-tuning sLLMs is the preferable strategy over prompting bLLMs.

Looking ahead, our future work will explore the versatile applications of bLLMs to enhance their efficacy in the SA4SE task. Meanwhile, we plan to apply the latest advancements in transformer accelartion [17, 18] to speed up the training and inference of bLLMs. We also plan to leverage the results of SA4SE for other downstream tasks, such as library or API recommendations [23, 70].

Replication Package. We release the data, code and results on: https://github.com/soarsmu/LLM4SA4SE.

REFERENCES

[1] Toufique Ahmed, Amiangshu Bosu, Anindya Iqbal, and Shahram Rahimi. 2017. SentiCR: A customized sentiment analysis tool for code review interactions. In *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 106–111.

[2] Toufique Ahmed, Kunal Suresh Pai, Premkumar Devanbu, and Earl Barr. 2024. Automatic semantic augmentation of language model prompts (for code summarization). In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. 1–13.

[3] Eeshita Biswas, Mehmet Efruz Karabulut, Lori Pollock, and K Vijay-Shanker. 2020. Achieving reliable sentiment analysis in the software engineering domain using bert. In *2020 IEEE International conference on software maintenance and evolution (ICSME)*. IEEE, 162–173.

[4] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258* (2021).

[5] Leo Breiman. 2001. Random forests. *Machine learning* 45 (2001), 5–32.

[6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[7] Fabio Calefato, Filippo Lanubile, Federico Maiorano, and Nicole Novielli. 2018. Sentiment polarity detection for software development. In *Proceedings of the 40th International Conference on Software Engineering*. 128–128.

[8] Fabio Calefato, Filippo Lanubile, Nicole Novielli, and Luigi Quaranta. 2019. Emtk-the emotion mining toolkit. In *2019 IEEE/ACM 4th International Workshop on Emotion Awareness in Software Engineering (SEmotion)*. IEEE, 34–37.

[9] Ning Chen, Steven CH Hoi, Shaohua Li, and Xiaokui Xiao. 2015. SimApp: A framework for detecting similar mobile applications by online kernel learning. In *Proceedings of the eighth ACM international conference on web search and data mining*. 305–314.

[10] Zhenpeng Chen, Yanbin Cao, Xuan Lu, Qiaozhu Mei, and Xuanzhe Liu. 2019. Sentimoji: an emoji-powered learning approach for sentiment analysis in software engineering. In *Proceedings of the 2019 27th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering*. 841–852.

[11] Zhenpeng Chen, Yanbin Cao, Huihan Yao, Xuan Lu, Xin Peng, Hong Mei, and Xuanzhe Liu. 2021. Emoji-powered sentiment and emotion detection from software developers' communication data. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 30, 2 (2021), 1–48.

[12] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality. https://lmsys.org/blog/2023-03-30-vicuna/

[13] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311* (2022).

[14] Xiang Deng, Vasilisa Bashlovkina, Feng Han, Simon Baumgartner, and Michael Bendersky. 2023. LLMs to the Moon? Reddit Market Sentiment Analysis with Large Language Models. In *Companion Proceedings of the ACM Web Conference 2023*. 1014–1019.

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL-HLT*. 4171–4186.

[16] Zulfadzli Drus and Haliyana Khalid. 2019. Sentiment analysis in social media and its application: Systematic literature review. *Procedia Computer Science* 161 (2019), 707–714.

[17] Cunxiao Du, Jing Jiang, Xu Yuanchen, Jiawei Wu, Sicheng Yu, Yongqi Li, Shenggui Li, Kai Xu, Liqiang Nie, Zhaopeng Tu, et al. [n. d.]. GliDe with a CaPE: A Low-Hassle Method to Accelerate Speculative Decoding. In *Forty-first International Conference on Machine Learning*.

[18] Cunxiao Du, Zhaopeng Tu, and Jing Jiang. 2021. Order-agnostic cross entropy for non-autoregressive machine translation. In *International conference on machine learning*. PMLR, 2849–2859.

[19] Tom Fawcett. 2006. An introduction to ROC analysis. *Pattern recognition letters* 27, 8 (2006), 861–874.

[20] Sidong Feng and Chunyang Chen. 2024. Prompting Is All You Need: Automated Android Bug Replay with Large Language Models. In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering*. 1–13.

[21] Mingyang Geng, Shangwen Wang, Dezun Dong, Haotian Wang, Ge Li, Zhi Jin, Xiaoguang Mao, and Xiangke Liao. 2024. Large language models are few-shot summarizers: Multi-intent comment generation via in-context learning. In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering*. 1–13.

[22] Mia Mohammad Imran, Yashasvi Jain, Preetha Chatterjee, and Kostadin Damevski. 2022. Data Augmentation for Improving Emotion Recognition in Software Engineering Communication. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*. 1–13.

[23] Ivana Clairine Irsan, Ting Zhang, Ferdian Thung, Kisub Kim, and David Lo. 2023. Multi-modal api recommendation. In *2023 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 272–283.

[24] Md Rakibul Islam, Md Kauser Ahmmed, and Minhaz F Zibran. 2019. MarValous: Machine learning based detection of emotions in the valence-arousal space in software engineering text. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*. 1786–1793.

[25] Md Rakibul Islam and Minhaz F Zibran. 2018. DEVA: sensing emotions in the valence arousal space in software engineering text. In *Proceedings of the 33rd annual ACM symposium on applied computing*. 1536–1543.

[26] Md Rakibul Islam and Minhaz F Zibran. 2018. SentiStrength-SE: Exploiting domain specificity for improved sentiment analysis in software engineering text. *Journal of Systems and Software* 145 (2018), 125–146.

[27] Robbert Jongeling, Subhajit Datta, and Alexander Serebrenik. 2015. Choosing your weapons: On sentiment analysis tools for software engineering research. In *2015 IEEE international conference on software maintenance and evolution (ICSME)*. IEEE, 531–535.

[28] Robbert Jongeling, Proshanta Sarkar, Subhajit Datta, and Alexander Serebrenik. 2017. On negative results when using sentiment analysis tools for software engineering research. *Empirical Software Engineering* 22 (2017), 2543–2584.

[29] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. (2020).

[30] Bin Lin, Nathan Cassee, Alexander Serebrenik, Gabriele Bavota, Nicole Novielli, and Michele Lanza. 2022. Opinion mining for software development: a systematic literature review. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 31, 3 (2022), 1–41.

[31] Bin Lin, Fiorella Zampetti, Gabriele Bavota, Massimiliano Di Penta, Michele Lanza, and Rocco Oliveto. 2018. Sentiment analysis for software engineering: How far can we go?. In *Proceedings of the 40th international conference on software engineering*. 94–104.

[32] Bing Liu. 2020. *Sentiment analysis: Mining opinions, sentiments, and emotions*. Cambridge university press.

[33] Bing Liu et al. 2010. Sentiment analysis and subjectivity. *Handbook of natural language processing* 2, 2010 (2010), 627–666.

[34] Bing Liu and Lei Zhang. 2012. A survey of opinion mining and sentiment analysis. In *Mining text data*. Springer, 415–463.

[35] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).

[36] Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically Ordered Prompts and Where to Find Them: Overcoming Few-Shot Prompt Order Sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 8086–8098.

[37] Zeyang Ma, An Ran Chen, Dong Jae Kim, Tse-Hsun Chen, and Shaowei Wang. 2024. Llmparser: An exploratory study on using large language models for log parsing. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. 1–13.

[38] Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the Role of Demonstrations: What Makes In-Context Learning Work?. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. 11048–11064.

[39] Moran Mizrahi, Guy Kaplan, Dan Malkin, Rotem Dror, Dafna Shahaf, and Gabriel Stanovsky. 2023. State of what art? a call for multi-prompt llm evaluation. *arXiv preprint arXiv:2401.00595* (2023).

[40] Alessandro Murgia, Marco Ortu, Parastou Tourani, Bram Adams, and Serge Demeyer. 2018. An exploratory qualitative and quantitative analysis of emotions in issue report comments of open source systems. *Empirical Software Engineering* 23 (2018), 521–564.

[41] Nicole Novielli, Fabio Calefato, Davide Dongiovanni, Daniela Girardi, and Filippo Lanubile. 2020. Can we use se-specific sentiment analysis tools in a cross-platform setting?. In *Proceedings of the 17th International Conference on Mining Software Repositories*. 158–168.

[42] Nicole Novielli, Fabio Calefato, Filippo Lanubile, and Alexander Serebrenik. 2021. Assessment of off-the-shelf SE-specific sentiment analysis tools: An extended replication study. *Empirical Software Engineering* 26, 4 (2021), 77.

[43] Nicole Novielli, Daniela Girardi, and Filippo Lanubile. 2018. A benchmark study on sentiment analysis for software engineering research. In *Proceedings of the 15th International Conference on Mining Software Repositories*. 364–375.

[44] Martin Obaidi, Lukas Nagel, Alexander Specht, and Jil Klünder. 2022. Sentiment analysis tools in software engineering: A systematic mapping study. *Information and Software Technology* (2022), 107018.

[45] Marco Ortu, Alessandro Murgia, Giuseppe Destefanis, Parastou Tourani, Roberto Tonelli, Michele Marchesi, and Bram Adams. 2016. The emotional side of software developers in JIRA. In *Proceedings of the 13th international conference on mining software repositories*. 480–483.

[46] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[47] Ethan Perez, Douwe Kiela, and Kyunghyun Cho. 2021. True few-shot learning with language models. *Advances in neural information processing systems* 34 (2021), 11054–11070.

[48] Anuja Priyam, Gupta R Abhijeeta, Anju Rathee, and Saurabh Srivastava. 2013. Comparative analysis of decision tree classification algorithms. *International Journal of current engineering and technology* 3, 2 (2013), 334–337.

[49] Laria Reynolds and Kyle McDonell. 2021. Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–7.

[50] Irina Rish et al. 2001. An empirical study of the naive Bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, Vol. 3. 41–46.

[51] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108* (2019).

[52] Klaus R Scherer, Tanja Wranik, Janique Sangsue, Véronique Tran, and Ursula Scherer. 2004. Emotions in everyday life: Probability of occurrence, risk factors, appraisal and reaction patterns. *Social Science Information* 43, 4 (2004), 499–570.

[53] Anna Schmidt and Michael Wiegand. 2017. A survey on hate speech detection using natural language processing. In *Proceedings of the fifth international workshop on natural language processing for social media*. 1–10.

[54] Noam Shazeer. 2020. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202* (2020).

[55] Nan Song, Hongjie Cai, Rui Xia, Jianfei Yu, Zhen Wu, and Xinyu Dai. 2023. A Sequence-to-Structure Approach to Document-level Targeted Sentiment Analysis. In *Findings of the Association for Computational Linguistics: EMNLP 2023*.

7687–7698.

[56] Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. 2021. Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864* (2021).

[57] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford Alpaca: An Instruction-following LLaMA model. https://github.com/tatsu-lab/stanford_alpaca.

[58] Parastou Tourani, Yujuan Jiang, and Bram Adams. 2014. Monitoring sentiment in open source mailing lists: exploratory study on the apache ecosystem.. In *CASCON*, Vol. 14. 34–44.

[59] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).

[60] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288* (2023).

[61] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[62] Lorenzo Villarroel, Gabriele Bavota, Barbara Russo, Rocco Oliveto, and Massimiliano Di Penta. 2016. Release planning of mobile apps based on user reviews. In *Proceedings of the 38th International Conference on Software Engineering*. 14–24.

[63] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*. 38–45.

[64] Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244* (2023).

[65] Junjielong Xu, Ziang Cui, Yuan Zhao, Xu Zhang, Shilin He, Pinjia He, Liqun Li, Yu Kang, Qingwei Lin, Yingnong Dang, et al. 2024. UniLog: Automatic Logging via LLM and In-Context Learning. In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering*. 1–12.

[66] Yiyan Xu, Wenjie Wang, Fuli Feng, Yunshan Ma, Jizhi Zhang, and Xiangnan He. 2024. Diffusion Models for Generative Outfit Recommendation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1350–1359.

[67] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems* 32 (2019).

[68] Peiyan Zhang, Yuchen Yan, Xi Zhang, Liying Kang, Chaozhuo Li, Feiran Huang, Senzhang Wang, and Sunghun Kim. 2024. GPT4Rec: Graph Prompt Tuning for Streaming Recommendation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1774–1784.

[69] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068* (2022).

[70] Ting Zhang, Divya Prabha Chandrasekaran, Ferdian Thung, and David Lo. 2022. Benchmarking library recognition in tweets. In *Proceedings of the 30th IEEE/ACM International Conference on Program Comprehension*. 343–353.

[71] Ting Zhang, Bowen Xu, Ferdian Thung, Stefanus Agus Haryono, David Lo, and Lingxiao Jiang. 2020. 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, 70–80.

[72] Wenxuan Zhang, Yue Deng, Bing Liu, Sinno Jialin Pan, and Lidong Bing. 2023. Sentiment Analysis in the Era of Large Language Models: A Reality Check. *arXiv preprint arXiv:2305.15005* (2023).

[73] Wenxuan Zhang, Xin Li, Yang Deng, Lidong Bing, and Wai Lam. 2022. A survey on aspect-based sentiment analysis: Tasks, methods, and challenges. *IEEE Transactions on Knowledge and Data Engineering* (2022).

[74] Yingying Zhang and Daqing Hou. 2013. Extracting problematic API features from forum discussions. In *2013 21st International Conference on Program Comprehension (ICPC)*. IEEE, 142–151.

[75] Xin Zhou, Ting Zhang, and David Lo. 2024. Large Language Model for Vulnerability Detection: Emerging Results and Future Directions. In *Proceedings of the 2024 ACM/IEEE 44th International Conference on Software Engineering: New Ideas and Emerging Results* (Lisbon, Portugal) *(ICSE-NIER'24)*. Association for Computing Machinery, New York, NY, USA, 47–51. https://doi.org/10.1145/3639476.3639762