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 is an inherently collaborative process, where various stakeholders frequently express their opinions and emotions across diverse platforms. Recognizing the sentiments conveyed in these interactions is crucial for the effective development and ongoing maintenance of software systems. Over the years, many tools have been proposed to aid in sentiment analysis, but accurately identifying the sentiments expressed in software engineering datasets remains challenging.

Although fine-tuned smaller large language models (sLLMs) have shown potential in handling software engineering tasks, they struggle with the shortage of labeled data. With the emergence of bigger large language models (bLLMs), it is pertinent to investigate whether they can handle this challenge in the context of sentiment analysis for software engineering. In this work, we undertake a comprehensive empirical study using five established datasets. We assess the performance of three open-source bLLMs in both zero-shot and few-shot scenarios. Additionally, we compare them with fine-tuned sLLMs.

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 [30]. 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 [17]. Generally speaking, SA contains several tasks, such as sentiment classification [28], aspect-based sentiment classification [63], or hate speech detection [45]. 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 [27] and identifying sentences conveying negative opinions about application programming interfaces

Authors' addresses: Ting Zhang, Singapore Management University, Singapore, tingzhang.2019@phdcs.smu.edu.sg; Ivana Clairine Irsan, Singapore Management University, Singapore, ivanairsan@smu.edu.sg; Ferdian Thung, Singapore Management University, Singapore, ferdianthung@smu.edu.sg; David Lo, Singapore Management University, Singapore, davidlo@smu.edu.sg.

0004-5411/2023/10-ART1 $15.00
https://doi.org/XXXXXXX.XXXXXXX

(APIs) [64]. 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 [23, 49]. This discrepancy has spurred a growing interest in developing SE-specific SA tools over the past decade [2, 11, 22]. These sentiment analysis for SE (SA4SE) tools usually either propose a SE-specific lexicon [22] or a SE-specific model [2]. At the same time, several benchmarking studies on evaluating general SA tools and SE-specific tools have been conducted [23, 27, 37, 61]. Zhang et al. [61] 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, 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 [35], 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 [50] 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) [5]. 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 [14]. bLLMs can make predictions conditioned on a few input-output examples without updating model parameters and achieve success in various tasks [6, 14]. 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 implications of our findings and the threats to validity. Section 6 shows the related works on SA4SE. Section 7 concludes this work and discusses future work.

2 BACKGROUND

2.1 Smaller Large Language Models in SE

Ever since the introduction of the attention mechanism [52], the Transformer architecture has been widely adopted in many NLP and SE tasks. These sLLMs are trained on broad data and are pre-trained with self-supervised learning. They can be further fine-tuned with domain-specific data on downstream tasks to achieve better performance. They are usually encoder-only or encoder-decoder architectures. For instance, BERT [16], which is based on an encoder and learns bidirectional encoder representations from Transformers, adopts a masked language model (MLM) to pre-train the model. Specifically, BERT randomly masks some tokens in the input, and the model is trained to predict the masked tokens.

Notably, many SE artifacts are written in natural language. Thus, encoder-based models, such as BERT [16] and RoBERTa [31] have been used to represent these kinds of artifacts better. For instance, RoBERTa has been used to encode Stack Overflow posts to enhance API recommendation [54]. Encoder-decoder-based models, such as T5 [41], have been leveraged to fix code errors [3] or repair vulnerabilities [18].

2.2 Bigger Large Language Models in SE

As the pioneer of bLLMs, GPT-3 [6] is based on decoder-only architecture and it shows that large autoregressive language models can achieve strong performance on many NLP tasks with only a few examples. These types of bLLMs are usually trained with a standard left-to-right language modeling objective: predicting the next token given the previous tokens. With the rise of bLLMs, the focus of NLP has shifted from the fine-tuning paradigm to the prompt engineering paradigm [5]. Recently, many large language models have been proposed, such as GPT-3 [6], PaLM [14], and LLaMA [50]. Among them, ChatGPT [1] has been widely used both in practice and in research. There have been many studies on evaluating or leveraging ChatGPT for various SE tasks, such as program repair [56], debugging [42], and duplicate bug report detection [60]. Most of the existing studies focus on the effectiveness of ChatGPT. However, since ChatGPT is closed-source and is still evolving, we consider that reporting the results from one version of ChatGPT may have limited effects in the future version of ChatGPT. Just over a few months, both GPT-3.5 and GPT-4 have been found to significantly change their "behavior", with the accuracy of their responses appearing to go down [9]. In addition, users need to pay for the official ChatGPT API to get reliable results. Using the ChatGPT graphical UI would lead to different results. After the introduction of ChatGPT, *Prompt 0*

Input: A chat between a curious user and an artificial intelligence assistant. The assistant gives helpful, detailed, and polite answers to the user's questions.

USER: *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.*

ASSISTANT: Sure!</s>

Sentence: its nice this apps is must lovely

Output: ASSISTANT: positive

Prompt 1

Input: A chat between a curious user and an artificial intelligence assistant. The assistant gives helpful, detailed, and polite answers to the user's questions.

USER: *Please categorize the sentiment expressed in the following sentence from APP reviews as either (1) positive, (2) neutral, or (3) negative.*

ASSISTANT: Sure!</s>

Sentence: its nice this apps is must lovely

Output: ASSISTANT: positive

Prompt 2

Input: A chat between a curious user and an artificial intelligence assistant. The assistant gives helpful, detailed, and polite answers to the user's questions.

USER: *I will give you a sentence from APP reviews. You need to reply with the sentiment expressed in the following sentence, either (1) positive, (2) neutral, or (3) negative.*

ASSISTANT: Sure!</s>

Sentence: its nice this apps is must lovely

Output: ASSISTANT: positive

Fig. 1. Zero-shot prompt template used by Vicuna [13] and WizardLM [57].

many open-source bLLMs have been proposed, such as Llama 2-Chat [51], Vicuna [13], and WizardLM [57]. The usage of these open-source bLLMs is still limited in the SE domain.

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 [32, 40]. Additionally, certain research findings have emphasized the substantial impact of different word orders within the prompt templates on the predictions [15, 33]. 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

[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`.

```
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 [51].

in their syntactic structure. Our inspiration for the first prompt template (i.e., Prompt 0) draws from Zhang et al. [62]. They designed the prompt to include only essential components, namely the task name, task definition, and output format. For Prompt 1 and Prompt 2, we introduce slight variations in expression.

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

RQ3: *How do prompting bLLMs compare with fine-tuned sLLMs on the SA4SE task?*
We compare the best macro-F1 and micro-F1 obtained by prompting bLLMs and the fine-tuned sLLMs.

3.2 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`, `GitHub`, `GooglePlay`, `Jira`, and `StackOverflow`.

`Gerrit` Dataset: Ahmed et al. [2] 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. [35] 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. [27] provided three datasets, each with its unique characteristics:

- **`GooglePlay` Dataset**: Originally collected by Chen et al. [10], this dataset contains user reviews of Android apps on Google Play. Villarroel et al. [53] 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. [39]. 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. The manual annotation was carried out individually by two annotators, 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 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

[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

and Jira, respectively. While some document units in these datasets contain multiple sentences and others just one, we collectively refer to them as *documents*.

3.3 Experimented 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 [55].

- **Llama 2-Chat** (`meta-llama/Llama-2-13b-chat-hf`) [51] is a fine-tuned version of Llama 2 that is optimized for dialogue use cases. Llama 2 uses the standard Transformer architecture [52] and it applies pre-normalization with RMSNorm [59], the SwiGLU activation function [46], and rotary positional embeddings [47]. 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`) [13] 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 [48].
- **WizardLM** (`WizardLM/WizardLM-13B-V1.2`) [57] 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 experimented in Zhang et al. [61], i.e., BERT [16], RoBERTa [31], ALBERT [25], XLNet [58]. 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 [55] in the parenthesis.

- **BERT** (`bert-base-uncased`) [16], which stands for Bidirectional Encoder Representations from Transformers, introduces two key pre-training tasks. The first is mask language modeling (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`) [31] is short for "A Robustly Optimized BERT Pretraining Approach". RoBERTa is a BERT variant distinguished by its innovative training strategies and

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

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`) [25], 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`) [58] 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`) [43] 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.

3.4 Evaluation Metrics

Following the prior works, we also use 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 [35].

3.5 Implementation Details

We run bLLMs and sLLMs on a machine with four NVIDIA RTX A5000. For bLLMs, we run each model with four GPUs. For sLLMs, we fine-tune each model with one GPU. 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.

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 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.

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 adopts 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

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

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%

bLLMs when employed with different prompt templates. Furthermore, it is noteworthy that even 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 17 instances. Following closely, *Prompt 1* leads in 13 instances. Interestingly, *Prompt 2*, while achieving the top performance on only 9 occasions, occasionally surpasses *Prompt 1* by a significant margin, notably in the case of the Llama 2-Chat within the `StackOverflow`

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	0.6	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	0.6	0.94	0.86	0.51
		5	0.77	0.6	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	0.7	0.91	0.92	0.74

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. These results show that Prompt 0 can achieve overall best results considering both macro-F1 and micro-F1.

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 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.

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%, and the average micro-F1 score difference ranges from 1.5% to 18.7%.

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.

In summary, when considering macro-F1, the 5-shot configuration emerges as the leader in seven instances, followed by the 3-shot setup in five cases, and the 1-shot configuration excels in four instances. Regarding micro-F1, the 5-shot configuration is at the top seven times, the 3-shot setup prevails nine 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. Given all the bLLMs benefit from more shots on the `Jira` dataset, we investigate the potential reason. One unique characteristic of the `Jira` dataset is that it is a binary-class dataset. Despite specifying the available label list as [*'negative'*, *'positive'*] in the prompt templates, bLLMs tend to predict the sentiment as *neutral*. We examined the cases where bLLMs predicted the sentiment as *neutral*:

In the zero-shot setting, Vicuna explicitly predicts the *neutral* label 4, 19, and 9 times with *Prompt 0, 1, 2*, respectively. Similarly, Llama 2-Chat deviated from the template to some extent but performed better than Vicuna, predicting the *neutral* label 1, 14, and 5 times. WizardLM, on the other hand, demonstrated a stronger adherence to the template by predicting *neutral* only 1, 0, and 1 times in the same scenarios.

This issue was mitigated when we introduced more shots under the few-shot setting: In the case of Vicuna, it explicitly predicted *neutral* 11, 8, and 5 times with 1-shot, 3-shot, and 5-shot prompt templates, respectively. Like zero-shot learning, WizardLM also predicted *neutral* 1, 0, and 1 times. The occurrences of Llama 2-Chat predicting the *neutral* label under few-shot learning were comparatively lower. With 1-shot, 3-shot, and 5-shot prompt templates, the numbers were 4, 4, and 3, respectively. This shows the potential benefits brought by introducing more shots.

Other than benefits brought by more shots, we also notice the decline in macro-F1 with an increase in the number of examples is apparent, particularly 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.[62].

Figure 3 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. All six prompts examined in their study are considered. This figure reveals that the influence of different prompts on 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 4 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

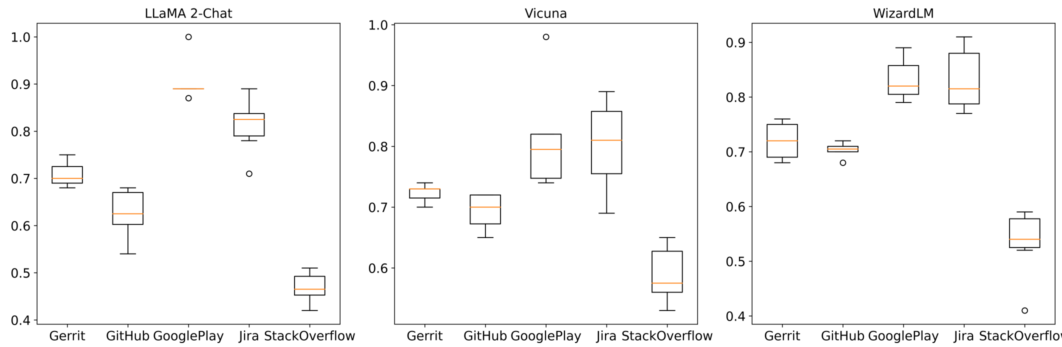


Fig. 3. Sensitivity of different prompt designs. Four different prompts give the performance variance of each dataset. The circles depicted in the figure represent outlier data points.

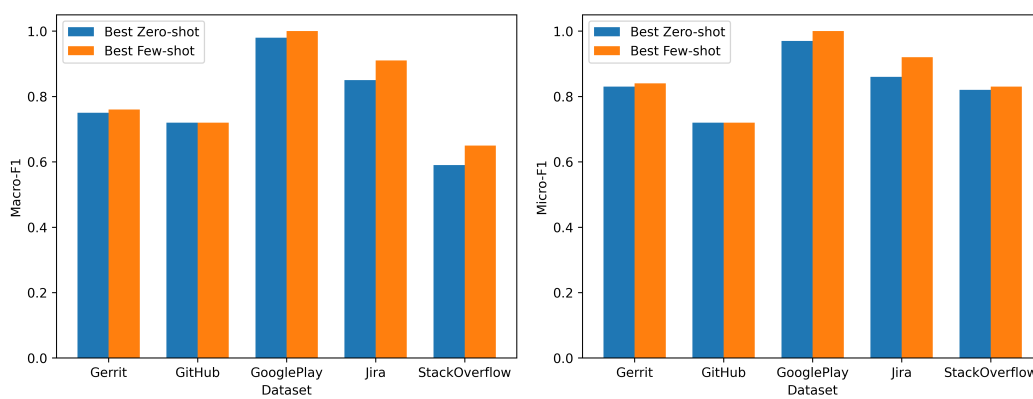


Fig. 4. Comparison of the highest macro-F1 and micro-F1 scores achieved through zero-shot learning and few-shot learning.

better under the few-shot learning paradigm. We conducted a Wilcoxon signed-rank test on two pairs of comparisons: the best zero-shot versus few-shot performance in terms of macro-F1 and the best zero-shot versus few-shot performance in terms of micro-F1. Notably, both comparisons yielded p-values of 0.07.

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 difference between the 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.

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	**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
	ALBERT	0.73	0.9	0.56	0.97	0.64
	BERT	0.75	0.92	0.49	0.95	0.57
	DistilBERT	0.81	0.92	0.57	0.95	0.6
	RoBERTa	0.74	0.94	0.42	0.95	0.68
	XLNet	0.77	0.91	0.39	0.94	0.67
Micro-F1	**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
	ALBERT	0.81	0.9	0.8	0.97	0.84
	BERT	0.8	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

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 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.

Moreover, in Figure 5, 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.

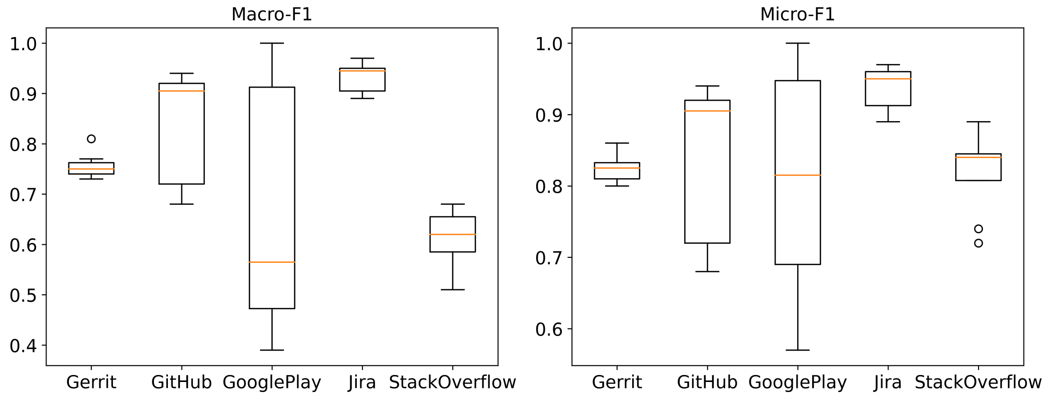


Fig. 5. Performance variance of all the models on each dataset.

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.

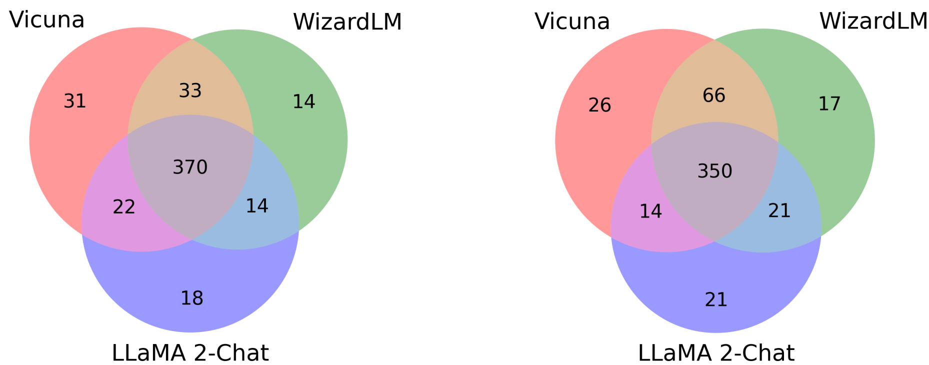
Answer to RQ3: In scenarios with limited labeled data and pronounced class imbalance, prompting bLLMs is a more effective strategy, outperforming fine-tuning sLLMs, e.g., in the `GooglePlay` dataset, Llama 2-Chat outperforms the DistilBERT by 75.4%. In contrast, when ample training data is available 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 Error Analysis

In this section, we conducted a quantitative and qualitative analysis to understand the main cause of misclassification made by bLLMs. In the first part, we choose the results achieved by the best-performing templates for analysis, i.e., for zero-shot learning, we analyze the results by *Prompt 0*; for few-shot learning, we analyze the results by *5-shot* prompt.

In Figure 6, 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 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



(a) Zero-shot learning with Prompt 0

(b) Few-shot learning with 5-shot prompt

Fig. 6. 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

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, all the bLLMs did not get any correct prediction no matter the setting. We adopt the error categories provided and used by Novielli et al. [35, 37]. Two authors (from here on, evaluators) first assigned a category to each document separately, which may not cover all the categories identified by Novielli et al. Afterward, they discussed and achieved agreement on the conflicts.

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 example, in `GitHub` and `StackOverflow` datasets, they were labeled as *neutral*. However, in `Gerrit` and `Jira` datasets, they

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%)

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 [29], 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 [44]. 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.

5 DISCUSSION

5.1 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.

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. In this context, manual crafting of prompt templates proves exceptionally advantageous, particularly in zero-shot scenarios. It is through this meticulous process that we can fully harness the capabilities of bLLMs. In the case of few-shot learning, dependent on the characteristics of the task, it may not always be helpful to add more shots. The longer context can confuse bLLMs and lead to worse results.

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.

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.

5.2 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. Additionally, we examined the influence of the number of shots in the few-shot setting. It is important to note that we based our prompt templates on prior work [62]. Another internal validity threat could be associated with potential data leakage. Nevertheless, the dataset we utilized cannot be directly accessed by visiting a webpage; it requires downloading from a specific URL. Consequently, the likelihood of data leakage is considered low. 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 3.2, 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.

6 RELATED WORK

In recent years, SA4SE has emerged as a vibrant and active research area within SE. This section provides an overview of the relevant literature, primarily focusing on the techniques proposed to boost SA4Se accuracy and empirical studies in SA4SE.

6.1 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 [2, 4, 7, 8, 11, 12, 19–22, 34].

Chen et al. [11] 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. [12] include an additional evaluation of SEntiMoji on the emotion detection task. They also compared it with four existing emotion detection methods, including DEVA [21], EmoTxt [8], MarValous [20], and ESEM-E [34]. 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, some research aimed at improving SA accuracy by handling existing challenges, such as labeled data scarcity. Imran et al. [19] 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. [61] 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.

6.2 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 [23, 24, 26, 27, 36–38].

Novielli et al., in their recent study [36], 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. [38] 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. (2022) investigated 185 papers on opinion mining in Software Engineering [26], 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.

7 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. We also plan to leverage the results of SA4SE for other downstream tasks, such as library or API recommendations.

REPLICATION PACKAGE

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

REFERENCES

[1] [n. d.]. ChatGPT. https://openai.com/chatgpt. (Accessed on 10/04/2023).

[2] 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.

[3] Berkay Berabi, Jingxuan He, Veselin Raychev, and Martin Vechev. 2021. Tfix: Learning to fix coding errors with a text-to-text transformer. In *International Conference on Machine Learning*. PMLR, 780–791.

[4] 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.

[5] 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).

[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] Lingjiao Chen, Matei Zaharia, and James Zou. 2023. How is ChatGPT's behavior changing over time? *arXiv preprint arXiv:2307.09009* (2023).

[10] 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.

[11] 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.

[12] 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.

[13] 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/

[14] 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).

[15] 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.

[16] 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.

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

[18] Michael Fu, Chakkrit Tantithamthavorn, Trung Le, Van Nguyen, and Dinh Phung. 2022. VulRepair: a T5-based automated software vulnerability repair. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 935–947.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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).

[26] 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.

[27] 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.

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

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

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

[31] 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).

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] 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.

[41] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research* 21, 1 (2020), 5485–5551.

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

[43] 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).

[44] 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.

[45] 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.

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

[47] 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).

[48] 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.

[49] 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.

[50] 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).

[51] 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).

[52] 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).

[53] 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.

[54] Moshi Wei, Nima Shiri Harzevili, Yuchao Huang, Junjie Wang, and Song Wang. 2022. Clear: contrastive learning for api recommendation. In *Proceedings of the 44th International Conference on Software Engineering*. 376–387.

[55] 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.

[56] Chunqiu Steven Xia and Lingming Zhang. 2023. Keep the Conversation Going: Fixing 162 out of 337 bugs for $0.42 each using ChatGPT. *arXiv preprint arXiv:2304.00385* (2023).

[57] 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).

[58] 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).

[59] 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).

[60] Ting Zhang, Ivana Clairine Irsan, Ferdian Thung, and David Lo. 2023. Cupid: Leveraging ChatGPT for More Accurate Duplicate Bug Report Detection. *arXiv preprint arXiv:2308.10022* (2023).

[61] 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.

[62] 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).

[63] 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).

[64] 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.