IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 26 June 2024, accepted 26 July 2024, date of publication 6 August 2024, date of current version 19 August 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3439450

RESEARCH ARTICLE

An Empirical Evaluation of the Zero-Shot, Few-Shot, and Traditional Fine-Tuning Based Pretrained Language Models for Sentiment Analysis in Software Engineering

MD SHAFIKUZZAMAN[1], MD RAKIBUL ISLAM[1], ALEX C. ROLLI[2], SHARMIN AKHTER[1], AND NAEEM SELIYA[2]

[1]Department of Computer Science, Lamar University, Beaumont, TX 77705, USA
[2]Department of Computer Science, University of Wisconsin–Eau Claire, Eau Claire, WI 54701, USA

Corresponding author: Md Rakibul Islam (mislam108@lamar.edu)

This work was supported in part by the College of Arts and Sciences, Lamar University, Beaumont, TX, USA.

ABSTRACT Recent advances in natural language processing (NLP) have led to the development of revolutionized pretrained language models (PLMs) impacting various NLP tasks, including sentiment analysis in software engineering. Choosing the right PLMs is crucial to effectively leverage these advanced PLMs. This paper presents the largest comparative evaluation of the PLMs for sentiment analysis in software engineering. Specifically, the study initially quantifies the performances of four traditionally fine-tuned PLMs, five zero-shot PLMs including `GPT-4` and `GPT-3` models, and three few-shot PLMs on six domain-specific datasets. The performances of the selected PLMs are also compared against two software engineering domain-specific traditionally fine-tuned PLMs and two state-of-the-art tools. The quantitative analysis reveals varying strengths across the different PLM types. The traditionally fine-tuned domain-specific PLM `seBERT` achieves the best results in the larger datasets, whereas the few-shot PLMs, such as `All-DistillRoBERTa`, show the best performances in the smaller datasets. A qualitative error analysis with the help of an Explainable AI technique uncovers existing challenges faced by PLMs in sentiment analysis for software engineering. The comprehensive quantitative and qualitative experiments significantly enrich knowledge in sentiment analysis in software engineering through reproducible insights.

INDEX TERMS Sentiment analysis, software engineering, natural language processing, pretrained language models, GPT-4, zero-shot learning, few-shot learning.

I. INTRODUCTION

Software development, being a collaborative process, is notably susceptible to the sentiment and emotion of developers [1], [2], [3], [4], [5]. Consequently, there has been a growing interest in sentiment analysis (SA) within the software engineering (SE) domain, aiming to detect and analyze positive, negative, and neutral sentiments of developers [1], [2], [3], [6], [7], [8], [9], [10], [11], [12], [13] to help to understand their impacts in SE.

As software development becomes more decentralized and collaborative, the ability to computationally understand sentiment in unstructured textual artifacts, such as comments/posts in developers' communications and interaction platforms, e.g., issue tracking system JIRA, source code repository GitHub, Q&A forum StackOverflow, and mailing lists, are growing in use [7], [8], [9], [12], [14] for SA. Initially, general-purpose SA tools (e.g., `SentiStrength` [15]) were applied for SA from the

The associate editor coordinating the review of this manuscript and approving it for publication was Loris Belcastro.

textual artifacts in SE. However, these tools, designed for broad applications, were reported to have poor accuracy in the SE [14], [16], [17]. Thus, researchers develop several general machine learning (ML) based customized tools, such as `Senti4SD` [18], `SentiCR` [19], and `EmoTxt` [10] to overcome the issue.

Recently, the Pretrained Large Language Models (PLMs), such as `BERT` [20] and `RoBERTa` [21] have significantly advanced various Natural Language Processing (NLP) tasks, including text summarizing, language translation, and sentiment classification - even extending their influences into SE [22], [23], [24], [25]. Zhang et al. [26] examine the performances of four PLMs, namely `BERT` [20], `RoBERTa` [21], `ALBERT` [27], and `XLNet` [28] for SA in SE and find that these PLMs outperform the existing SE domain-specific ML and ruled-based tools.

However, the aforementioned PLMs are required to be fine-tuned using the traditional method, where 70% to 80% of data is used for fine-tuning. From now on, we will call these traditionally fine-tuned PLMs as TPLMs. The requirement of huge training data for fine-tuning is exacerbated in domains, such as SE, where collecting and labeling sufficient training data is often expensive, time-consuming, and error-prone. Moreover, labeling data also requires substantial domain-specific knowledge. Such challenges have led to a limited number of datasets with an imbalanced distribution of sentiment types that may cause low performance for the TPLMs [24], [29] for SA in SE.

Further advances in NLP have brought us other advanced technologies, including zero-shot, few-shot, and bigger PLMs, e.g., `GPT-4` [29], [30] that are potential solutions to these issues. A zero-shot PLM (ZPLM) can classify labels without prior training [31]. Conversely, a few-shot PLM (FPLM) aims to classify labels by relying on a small set of illustrative examples rather than intensive training [32]. Moreover, bigger PLMs trained on massive corpora of texts and containing many parameters are zero-shot learners as they can be adapted to a downstream task simply by providing it with a prompt (a natural language description of a given task).

Nevertheless, the performance of the ZPLMs, FPLMs, and bigger PLMs, e.g., GPT-4 (considered as ZPLMs in this study) remains largely unexplored for SA in SE. Thus, in this study, we evaluate and compare the performances of ZPLMs and FPLMs against TPLMs. In particular, we ask the first research question (RQ1) as follows.

RQ1: *How do the ZPLMs and FPLMs perform compared to the TPLMs in detecting sentiments in the SE domain?*

To answer this question, we select five ZPLMs (including two bigger PLMs: `GPT-4` and `GPT-3.5`), five FPLMs, and four TPLMs. Then, we apply the selected 14 PLMs on six SE domain datasets (described in Section III-C) to quantitatively measure their performances in detecting sentiments. This will help one to choose the most appropriate PLMs among the 14 selected PLMs for SA in SE.

While all the selected PLMs in RQ1 are domain-independent, i.e., those PLMs are pretrained with general English corpus, the recent development of SE domain-specific PLMs makes us curious to examine their performances across diverse datasets. Moreover, we aim to compare the performances of state-of-the-art tools against the PLMs in the same settings. Thus, we ask the second research question (RQ2) as follows.

RQ2: *How do the domain-specific PLMs and state-of-the-art tools perform compared to the domain-independent PLMs in detecting sentiments in the SE domain?*

To answer this question, we select two domain-specific PLMs and two state-of-the-art tools: one rule-based tool, and one traditional machine learning-based tool. Then, we apply those PLMs and tools to the same datasets used for RQ1 to quantitatively measure and compare their performances against the best performing domain-independent PLMs. Such comparison provides us with the bigger picture to select the best options based on different scenarios.

Finally, to supplement our quantitative analysis, we also conduct a qualitative analysis to determine the dominant causes of the errors of the best-performing PLMs/tools for SA in SE. Thus, we ask the third research question (RQ3) as follows.

RQ3: *What are the dominant causes of the errors in sentiment prediction for the best-performing PLMs/tools?*

To answer this question, we employ two human raters and an *Explainable Artificial Intelligence* (XAI) tool, namely SHapley Additive exPlanations (`SHAP`) [33] to gain a deeper understanding of the causes of errors. The findings can help to further improve the performance of the PLMs/tools.

Contributions: The major contributions of this study are four-fold: (i) We are the first to apply the ZPLMs, including the `GPT-4` and `GPT-3` models, and FPLMs, for SA in the SE domain, (ii) we evaluate the performance of two domain-specific PLMs in a bigger settings for SA in SE, (iii) we for the first time conduct an error analysis of the with the help of an XAI tool to identify the errors for SA in SE, and (iv) to enable reproducibility, we publicly share the datasets and scripts used in our experiments [34].

II. BACKGROUND

Here, we describe the terminology that helps a reader to understand the content of the paper. **Traditional Fine-tuning (TFT)** is the most prevalent method to adapt a PLM by training it on a task-specific dataset with thousands of labeled examples. It excels in achieving high performance on various benchmarks. However, it demands a new sizable dataset for each task and might lack generalization beyond its original data [29].

To overcome the issues with TFT, **Zero-shot Learning (ZSL) and Few-shot Learning (FSL)** methods are devised that require zero to a limited amount of training data. For example, ZSL aims to recognize new classes without any labeled examples. This is achieved by leveraging knowledge

transfer between seen and unseen classes using semantic embeddings, hierarchical relationships, or generative models [31]. ZSL has recently been adapted to many NLP tasks, including entity recognition [35], relation extraction [36], and text classification [37].

Conversely, FSL involves training models using a minimal amount of labeled data. It is a more supervised approach compared to ZSL but requires significantly fewer labeled examples than traditional supervised learning. Models are trained to generalize from this limited labeled data, often through meta-learning, metric-based methods, data augmentation, and transfer learning [32].

Natural Language Inference (NLI) also known as textual entailment that determines the relationship between a premise text p and hypothesis text h [38]. The goal is to predict a label y, where $y \in \{entailment, contradiction, neutral\}$.

If the meaning of the hypothesis can be inferred or logically deduced from the premise, it is considered an entailment. For example, as shown in Table 1, for the given first premise, "*The cat is sitting on a chair*", the first hypothesis - "*The cat is on a piece of furniture*" - can be logically inferred. Again, if the meaning of the hypothesis contradicts or is in direct opposition to the premise, it is considered a contradiction. Finally, if the relationship between the premise and the hypothesis is neither entailment nor contradiction, it is considered neutral. The second and third rows in Table 1 show premises and hypotheses categorized as contradicting and neutral.

NLI helps capture the complex relationships between sentences, providing insights into the logical connections, implications, and contradictions within textual content. Many NLP applications, such as text classification, question answering, information extraction, and summarization, must recognize that a particular target meaning can be inferred from different text variants. Thus, many PLMs, e.g., `BERT`, and `RoBERTa` that are fined-tuned using NLI datasets, e.g., Multi-Genre Natural Language Inference (MNLI) [39] show superior results in various NLP tasks.

Sentence BERT (`SBERT`) [40] is a family of the PLMs that are specifically optimized for generating sentence embeddings. They are trained on NLI datasets, such as MNLI, to encode sentences into fixed-sized vectors so that semantically similar sentences are close together in the embedding space. This allows using `SBERT` to efficiently find similar sentences by computing *cosine similarity* between the dense embeddings. The key benefits of `SBERT` include the ability to derive high-quality sentence embeddings for downstream tasks, such as semantic search, text classification, and clustering. The pretrained contextual representations also transfer well to new data. There is a Python framework [41] available that offers an extensive collection of `SBERT` models fine-tuned using NLI datasets to generate FPLMs, such as `LaBSE` [42] and `All-MPNET-base-v2` [43], [44].

SetFit [45] works by first fine-tuning a pretrained `SBERT` on a small number of text pairs in a contrastive Siamese manner. The resulting model is then used to generate rich text embeddings, which are used to train a classification head. This simple framework requires no prompts or verbalizers and achieves high accuracy with orders of magnitude fewer parameters than existing techniques.

TABLE 1. Categories of Hypothesis in NLI.

Premise (t)	Hypothesis (h)	Label
The cat is sitting on a chair	The cat is on a piece of furniture	Entailment
A black race car starts up in front of a crowd of people	A person is driving down a lonely road	Contradiction
A person inspects the uniform of a figure	The person is playing tennis	Neutral

Self-training [46] provides a simple yet effective semi-supervised approach to improve the performance of ZSL for downstream tasks. The self-training process requires only unlabeled data from the target domain. This technique can be realized for text classification by leveraging NLI datasets. Concretely, a labeled NLI corpus, such as MNLI, is first utilized to train a textual entailment model f_θ. This model is then applied to unlabeled text from a different target distribution X_t, such as texts from the SE domain. Given sentences are classified by f_θ as entailment or contradiction with probability exceeding a confidence threshold τ are added to the training set as pseudo-labeled examples $\hat{Y}_t$:

$$\hat{Y}t = f_\theta(x)|x \in X_t, P(f_\theta(x)) > \tau$$

The entailment model is retrained on this expanded dataset $(X_t, \hat{Y}_t)$, and the self-training cycle repeats. After sufficient iterations, the resulting model gains coverage of the target domain lexicon and linguistic patterns. This enables zero-shot generalization without reliance on hand-crafted attributes or semantic lexicons. Empirically, self-training boosts performance over supervised NLI models according to prior work [24], [35], [47].

III. METHODOLOGY

In this Section, we briefly describe the PLMs, the state-of-the-art tools, the datasets, the evaluation metrics, and the experimental settings used to conduct the experiments.

A. DOMAIN-INDEPENDENT PRETRAINED LARGE LANGUAGE MODELS

1) TRADITIONAL PLM (TPLM)

In the following, we briefly describe the four domain-independent and two domain-specific TPLMs used in this study.

`BERT`: Bidirectional Encoder Representations from Transformers, aka, `BERT` is a bidirectional transformer-based model pretrained using masked language modeling and next sentence prediction objectives [20]. `BERT` uses the standard transformer encoder architecture, consisting of multiple layers of multi-head self-attention and feedforward layers. Through its masked language modeling pretraining, `BERT` learns deep bidirectional representations from unlabeled text by randomly masking input tokens and predicting the

masked words based on context. In addition to masked language modeling, BERT is pre-trained on next-sentence prediction to learn relationships between sentences. For pretraining, Devlin et al. [20] leveraged the *BooksCorpus* dataset (800 million words) as well as *English Wikipedia* (2,500 million words). Subword tokenization uses *WordPiece* to handle rare and unknown words. With pretrained representations encoding substantial language knowledge, BERT achieves strong performance on sentence-level downstream tasks with just an additional output layer during fine-tuning. BERT has two model sizes, $\text{BERT}_{\text{base}}$ and $\text{BERT}_{\text{large}}$, with 110 million and 340 million parameters, respectively. In this study, we use the $\text{BERT}_{\text{base}}$ model as it takes considerably less time to fine-tune for the downstream tasks.

RoBERTa. Robustly Optimized BERT Pretraining Approach, i.e., RoBERTa builds on top of the original BERT model by modifying key hyperparameters and training procedures [21]. Through a comprehensive study exploring the effects of critical hyperparameters and training data size, Liu et al. [21] identified improvements to BERT's pretraining methodology. Notably, RoBERTa is trained on larger mini-batches for more iterations over increased data. Furthermore, the next sentence prediction objective used during BERT pretraining is removed in RoBERTa, with the model being trained on longer input sequences instead. Additionally, RoBERTa incorporates dynamic masking, generating a new random masking pattern for each input sequence rather than using a static masked pattern. The enhanced pretrained representations from modifications to BERT's training enable significant performance gains on downstream tasks through fine-tuning.

XLNET. Transformer-XL Network (XLNET) is a model that employs a generalized autoregressive pretraining method to address the pretraining-finetuning discrepancy found in models, such as BERT. Instead of masking input tokens, XLNET maximizes the expected likelihood over all permutations of the input sequence factorization order using *Transformer-XL* as the backbone model architecture. This permutation language modeling objective allows XLNET to learn bidirectional contexts while retaining autoregressive modeling benefits. Additionally, XLNET utilizes the segment recurrence mechanism from *Transformer-XL* to model longer-term dependencies. These innovations enable XLNET to outperform BERT on various language understanding tasks, including question answering, natural language inference, and sentiment classification.

ALBERT. This lite version of BERT incorporates parameter reduction techniques to lower memory consumption and increase training speed [27]. ALBERT employs factorized embedding parameterization and cross-layer parameter sharing to significantly reduce the number of parameters from BERT. In addition, ALBERT is trained with sentence order prediction loss instead of next sentence prediction. With fewer parameters and other optimizations like a unified embedding matrix and repeating layers split across groups, ALBERT retains or improves upon BERT's performance while using substantially fewer resources. For example, ALBERT-base has 11M parameters compared to BERT-base's 110M parameters.

2) ZERO-SHOT PLM (ZPLM)

In the following, we describe the five ZPLMs used in this study.

BART-large-mnli (BART-mnli) [48]: This is a ZPLM that applies the pretrained BART (Bidirectional and Auto-Regressive Transformers) [49] model to natural language inference tasks by fine-tuning on the *MNLI* dataset [39]. BART uses a standard seq2seq transformer architecture with a bidirectional encoder and autoregressive decoder. The model is pretrained by reconstructing corrupted document-level texts, forcing it to learn bidirectional representations. $\text{BART}_{\text{large}}$ is a version of BART with over 400 million parameters. Fine-tuning this pretrained $\text{BART}_{\text{large}}$ model on the MNLI dataset provides task-specific adaptations for textual entailment and inference. MNLI contains sentence pairs with textual entailment annotations across different genres of written and spoken English. By leveraging the bidirectional pretraining of $\text{BART}_{\text{large}}$ and then fine-tuning it for inference using the MNLI dataset, BART-large-mnli achieves strong performance on question answering, textual entailment, and other natural language understanding tasks centered around reasoning about entailment relationships.

RoBERTa-large-mnli [50]. This ZPLM is developed by pretraining the RoBERTa model using the MNLI dataset. Thus, the architecture of the model is the same as the model RoBERTa.

DeBERTa-large-mnli-zero-cls (DeBERTa-mnli) [51]. This is another ZPLM based on the DeBERTa (Decoding-enhanced BERT with disentangled attention) architecture [52]. The DeBERTa architecture modifies BERT by disentangling the concept of self-attention and relative position encoding, which enables the model to incorporate absolute positional information better. This allows DeBERTa to learn more sophisticated syntactical or semantic relationships within the input text. The large version of DeBERTa has 24 transformer layers, a hidden size of 1024, and 24 attention heads, comprising over 400 million parameters in total. It was pretrained on masked language modeling using a large corpus of text data, then fine-tuned on the MNLI dataset for natural language inference to generate the Deberta-large-mnli model. In addition, the zero-cls variant of Deberta-large-mnli excludes the classification head during pretraining to avoid task-specific bias.

GPt-3.5 [53]: The bigger PLM GPT-3.5 is a variant of OpenAI's *GPT-3* [54] family, designed as a large-scale language model based on the transformer architecture. While OpenAI has not publicly disclosed the specific number of parameters for this model, it can range up to *175 billion* parameters as mentioned elsewhere [29]. The transformer architecture uses the *self-attention* mechanisms, which allows

the model to process and generate human-like text by considering the relationships between all words in a sentence, regardless of their position. As an *autoregressive* language model, GPT-3.5 predicts each subsequent word in a sequence by considering the words that precede it, making it capable of generating coherent and contextually relevant text across a wide range of topics and formats. Its training involved massive datasets from diverse internet text, enabling robust performance in natural language understanding and generation tasks.

GPT-4 [53]. This version of the GPT model was released by OpenAI in March 2023. It is a multimodal transformer model, A large-scale language model that accepts image and text inputs and produces text outputs. The model is trained using publicly available internet data and data licensed from third parties and then fine-tuned using Reinforcement Learning from Human Feedback (RLHF). It was compared with state-of-the-art models using Measuring Massive Multitask Language Understanding (MMLU) [55] that covers 57 tasks in elementary mathematics, US history, computer science, law, and more and outperformed them all.

3) FEW-SHOT PLM (FPLM)

In the following, we describe the FPLMs used in this study.

All-MiniLM-L6-v2 [56]: This FPLM is a distilled natural language understanding model based on the MiniLM architecture, which utilizes a multi-layer transformer network like BERT but with significantly fewer parameters. All-MiniLM-v2 has six layers, eight attention heads, and a hidden size 384, resulting in 14 million parameters [56]. This small size enables efficient inference while still achieving high performance on NLP tasks through knowledge distillation during pretraining.

All-MPNET-base-v2 (All-MPNET-base) [44]: This FPLM is developed by fine-tuning the Mpnet-base model [43] on one billion sentence pairs from several datasets [44] using contrastive learning objective [57]. The Mpnet-base (Masked and Permuted Pre-Training) model includes two new pretraining objectives - Masked Language Modeling (MLM) and Permuted Language Modeling (PELM). In MLM, some tokens are randomly masked, and the model is trained to predict them based on context, similar to BERT's masking strategy. In PELM, the order of sentences is randomly permuted, and the model must predict the original order, teaching the model about discourse coherence. By combining MLM and PLM during pre-training, MPNET can capture both local fluency and global coherence in text. Experimental results demonstrate that MPNET achieves state-of-the-art performance on several NLP benchmarks compared to previous models like BERT and RoBERTa.

All-DistilRoBERTa-v1 (All-DistilRoBERTa) [58]: This model is a distilled natural language understanding model based on the DistilRoBERTa architecture, which compresses RoBERTa into a smaller and faster variant using knowledge distillation during pretraining [59]. Specifically, All-DistilRoBERTa-v1 has six layers, 768 hidden dimensions, and 82 million parameters.

LaBSE [42]: Language-Agnostic BERT Sentence Embedding (LaBSE) is a pretrained multilingual sentence embedding model introduced by Feng et al. [42]. It is based on distilRoBERTa and trained on over 100 languages using a translated version of the BookCorpus dataset with only 110 million parameters, making it relatively lightweight compared to the monolingual BERT model [20] for each language. LaBSE generates 512-dimensional sentence embeddings that capture semantic similarity across different languages and outperforms previous multilingual models, such as LASER [60] and *multilingual* BERT [61] on cross-lingual semantic textual similarity tasks.

B. DOMAIN-SPECIFIC TPLMs AND STATE-OF-THE-ART TOOLS

1) TPLM

Here, we briefly describe the two domain-specific TPLMs used in this study.

seBERT [62]: This domain-specific TPLM is designed specifically for the software engineering (SE) domain. Unlike domain-independent TPLMs, seBERT is trained on a substantial corpus of SE-specific data, including over 119 GB data from sources, such as Stack Overflow posts, GitHub issues, Jira issues, and GitHub commit messages. This targeted pre-training allows seBERT to capture the unique vocabulary, terminologies, and contextual meanings prevalent in SE, which general language models often miss. seBERT has been validated through various SE-specific tasks, demonstrating superior performance in issue type prediction, commit intent prediction, and sentiment analysis within the SE domain.

BERT_SE [63]: This is another SE-specific BERT that is developed/pretrained using over 456K text items collected from Stack Stack Overflow posts and user requirements from 38 open-source projects. The assessment of BERT_SE is performed using the software requirements classification task, demonstrating that this model has an average improvement rate of 13% compared to the BERT-base model.

Table 2 shows the base architectures, number of parameters, layers, hidden layers, and head of the 15 PLMs used in this study.

2) STATE-OF-THE-ART TOOLS

Here, we describe the two state-of-the-art tools for SA in SE.

SentiCR [19]: SentiCR leverages the *Gradient Boosting Tree* (GBT) algorithm to classify two polarities, i.e., negative and non-negative. We re-train it on each dataset to classify three polarities, i.e., positive, neutral, and negative.

SentiStrength-SE: The tool SentistrengthSE [16] is the first domain-specific tool specially developed for sentiment analysis in software engineering text. Given piece of text T, SentiStrength-SE computes a pair $\langle p_c, n_c \rangle$ of integers, where $+1 \leq p_c \leq +5$ and $-5 \leq n_c \leq -1$. Here, p_c

TABLE 2. Architectures and specs of the PLMs.

Cat.	Model	Base	Param.	Layers	HL	Head
TPLM	BERT [20]	BERT-base	110M	12	768	12
	XLNeT [28]	XLNeT-base	117M	12	768	12
	RoBERTa [21]	RoBERTa-base	125M	12	768	12
	ALBERT [27]	ALBERT-base-v2	12M	12	768	12
	seBERT* [62]	BERT-large	340M	24	1,024	16
	BERT_SE* [63]	BERT-base	110M	12	768	12
ZPLM	BART-mnli [48] [64]	BART-large	406M	12	1,024	16
	RoBERTa-mnli [50]	RoBERTa-large	355M	24	1,024	16
	DeBERTa-mnli [51]	DeBERTa-large	355M	24	1,024	16
	GPT-4 [53]	GPT	U	U	U	U
	GPT-3.5 [65]	GPT	175B	U	U	U
FPLM	All-MiniLM-L6-v2 [66]	MiniLM-L6	66M	6	384	12
	All-mpnet-base [44]	MPNet-base	109M	12	768	12
	All-distilroberta [58]	DistilRoBERTa	110M	6	768	12
	LaBSE [42]					

*Domain-specific TPLM, U=**Unknown**, HL=Hidden Layers

TABLE 3. Summary of the datasets used in this study.

Data	#Doc	Sentiment Distribution			
		#Pos (%)	#Neg. (%)	#Neu. (%)	#NN. (%)
GIT	7,122	2013 (28.3)	2,087 (29.3)	3,022 (42.4)	NA
ARC	4,522	890 (19.7)	496 (11.0)	3,136 (69.3)	NA
SOC	1,500	131 (08.7)	178 (11.9)	1,191 (79.4)	NA
MAC	341	186 (54.5)	130 (38.1)	25 (07.3)	NA
JIC	926	290 (31.3)	636 (68.7)	NA	NA
CRC	1,600	NA	398 (24.9)	NA	1,202 (75.1)

NN.=Non-negative; NA=Not applicable

and n_c represent the positive and negative sentimental scores for the given text T.

C. DATASETS

This comparative study uses six publicly available SE domain-specific datasets with human-annotated polarity labels. Table 3 summarizes key statistics of the datasets, including the total number of document items in each dataset (#doc), and the specific number and percentage of documents for each sentiment labeled as #sentiment type (%). In the following, we describe the datasets briefly.

1) GitHub PULL REQUEST AND COMMIT COMMENTS (GIT) [9]

This dataset was developed by Novielli et al. and contains 7,122 pull requests and commit comments collected from GitHub. The dataset was annotated by three human raters where 28% of the comments convey positive sentiment, 29% express negative sentiment, and the remaining 43% are labeled as neutral.

2) API REVIEWS COMMENTS (ARC) [67]

The benchmark consisted of 4,522 sentences from 1,338 Stack Overflow posts. At least two coders manually labeled each sentence. In the dataset, the proportion of positive and negative sentiments are 19.7% and 11%, respectively, while 69.3% sentences are neutral.

3) STACK OVERFLOW POSTS (SOC) [11]

This dataset contains 1,500 sentences obtained from discussions tagged with Java in the Stack Overflow. Each sentence was manually classified with sentiment by two evaluators. The third evaluator was used to resolve any conflicting ratings. In the dataset, 8.7% sentences are labeled with positive sentiment, 11.9% sentences express negative sentiment, and the remaining 79.4% sentences are sentimentally neutral.

4) MOBILE APP REVIEW COMMENTS (MAC) [11]

To prepare the dataset, Lin et al. [11] randomly selected 341 samples from a collection of 3,000 app review comments prepared by Villarroel et al. [68]. Similar to the SOC dataset, two evaluators manually labeled the sentiment of each selected review, with a third evaluator to resolve any conflicts between the first two evaluators. Finally, the dataset contains 54.5% and 38.1% comments with positive and negative sentiment, respectively, while the remaining 7.3% comments are neutral.

5) JIRA ISSUE COMMENTS (JIC) [11]

This dataset was prepared by Ortu et al. [69] with sentences labeled with four emotions: love, joy, anger, and sadness. Later, Lin et al. [11] assigned positive polarity to sentences labeled with love and joy, and negative polarity to sentences labeled with anger and sadness. The dataset contains 926 samples with 31.3% positive and 68.7% negative sentiments.

6) CODE REVIEW COMMENTS (CRC) [19]

Ahmed et al. [19] developed this dataset that contains 1,600 code review comments. Three human annotators independently labeled each of the comments as positive, negative, or neutral. However, the positive and neutral comments were merged into a single non-negative class in the publicly released version of the dataset. Therefore, for this current work, we utilize the publicly available version of the dataset, in which 24.9% of the comments convey negative sentiment and the remaining 75.1% express non-negative (neutral and positive) sentiment.

D. EVALUATION METRICS

The performance of each PLM is measured in terms of *precision* (p), *recall* (r), and *F-score* ($F1$) metrics for each

of the sentimental polarities. Given a set T of texts, *precision* (p), *recall* (r), and *F-score* $F1$ for a particular sentimental polarity e is calculated as follows:

$$p = \frac{|\ T_e \cap T'_e\ |}{|\ T'_e\ |}, \quad r = \frac{|\ T_e \cap T'_e\ |}{|\ T_e\ |}, \quad F1 = \frac{2 \times p \times r}{p + r}$$

where T_e represents the set of texts having sentimental polarity e, and T'_e denotes the set of texts that are detected (by a PLM) to have the sentimental polarity e.

Moreover, to assess the overall performance of a PLM over a dataset, we compute *macro-average* for each metric $c \in p, r, F1$ as follows.

$$\mathcal{M}(c) = \frac{1}{k} \sum_{i=1}^{k} c_i$$

In the above equation, $\mathcal{M}(c)$ represents the macro-average of a metric c; c_i and n_i denote the metric c and number of texts for the i^{th} sentiment category (i.e., positive, negative, or neutral), respectively, while k represents the number of sentiment categories.

E. EXPERIMENTAL SETTING

In the following, we describe the experimental settings of this study.

1) GENERATING AND USING STRATIFIED 10-FOLD DATASET

To compute the performance of a selected PLMs, we apply *Repeated random sub-sampling cross validation* [70]. To do that, we use the following steps. (i) We randomly split the original data $\mathcal{D}$ into a stratified fold that contains two sets: (a) 80% of the data as training, and (b) the remaining 20% data for testing. (ii) We repeat first step k=10 times to create stratified 10-fold using `StratifiedShuffleSplit` [71]. (iii) The selected PLM is fine-tuned on the training set for a selected fold. (iv) Then, we test the PLM on the selected fold's test set to obtain its performance. (v) Steps iii and iv are repeated for each of the 10 folds. (vi) Finally, we average the performance metrics (of the ten runs) to get the final performance of the PLM on the dataset $\mathcal{D}$.

2) FINE-TUNING THE TPLMs

To perform sentiment classification, we add a feed-forward dense layer and softmax activation function on top of each TPLM. Then, we fine-tune each of the TPLMs and obtain their performance results using the cross-validation technique described above. We use four epochs with a batch size of 16 to run each TPLM. We set the learning rate to $2e-5$ and use the *AdamW* optimizer in the fine-tuning process.

3) FINE-TUNING THE FPLMs

We apply the *SetFit* [45] technique to fine-tune each FPLM. Instead of using 80% train data from a fold, for the FPLMs, we use randomly selected 50 examples from each sentiment category as the training data. Here, we use 10 epochs with a batch size of 16 and a learning rate of $2e-5$.

"role": "system"

"content": You are a sentiment analysis assistant.

"role": "user"

"content": I will give you a text collected from {dataset_name}. You classify the sentiment of the given text into: {sentiment types}. No explanation is needed for your output. Sentence:'<text>'

FIGURE 1. Prompt used in this study to instruct `GPT` models.

4) FINE-TUNING `GPT-3.5` WITH FEW-SHOT

We also use the model `GPT-3.5` in a few-shot setting. To do that, we fine-tune the `GPT-3.5` model by following the guidelines established by OpenAI. We employ a random sampling approach to select 50 data instances from each sentiment category within a dataset D. Subsequently, we use the OpenAI API[1] to facilitate fine-tuning of the model using randomly selected data instances. After completing the fine-tuning process, we obtain the fine-tuned model and name it as `GPT-3.5F`. Please note that at the time of this study, fine-tuning `GPT-4` was an experimental access program from OpenAI [92], and no guideline was available for this model. Thus, we avoid fine-tuning the `GPT-4` model.

5) FINE-TUNING THE ZPLMs

While ZPLMs are developed to be used in zero-shot learning settings, the performance of the ZPLMs can be improved by fine-tuning them [72]. Thus, we use both the base and fine-tuned ZPLMs in this study. To fine-tune the ZPLMs, we use the *self-training* technique devised by Gera et al. [72]. In contrast to the traditional fine-tuning process, the *self-training technique does require only unlabeled data.*

6) FINE-TUNING `SentiCR`

The state-of-the-art `SentiCR` is programmed in such a way that it takes a dataset and randomly creates the ten folds with training and testing sets in each fold. We have modified the code of `SentiCR` to stop its 10-fold data creation and feed our own created 10-fold training and testing sets for each dataset. Apart from that, we have used the default settings to fine-tune it.

7) PROMPTING FOR THE `GPT` MODELS

We follow the general guidelines provided by OpenAI[2] for prompt engineering. We design a prompt to instruct a `GPT` model as a sentiment analysis assistant for our sentiment classification task on a selected dataset. From a user's perspective, we provide the prompt as shown in Figure 1, for

[1] https://platform.openai.com/docs/api-reference
[2] https://platform.openai.com/docs/guides/prompt-engineering/strategy-write-clear-instructions

each GPT model to detect the sentiment expressed in given texts in a dataset. For each dataset, the variable *dataset_name* is replaced by the name of the dataset, and the variable *sentiment types* is replaced with the available sentiments of a dataset. We use the default temperature for each GPT model.

Please note that, SentiStrength-SE and zero-shot GPT-4 and GPT-3.5 do not require any fine-tuning, and their performances are computed using the test sets of 10-fold data for each dataset.

F. STATISTICAL TEST

The McNemar's statistical test [73], also known as the McNemar's test, is a widely employed non-parametric method for evaluating the statistical significance of performance differences between two classification models, such as sentiment classifiers. We employ McNemar's test to address this issue to determine whether the observed performance gap between the two methods is statistically significant. In the context of sentiment analysis research, the McNemar's test is a well-suited statistical method for our purposes as it does not rely on the assumption of data normality, which is often violated in sentiment analysis datasets. Furthermore, the widespread adoption of the McNemar's test in related research [16] on sentiment analysis models lends credibility and consistency to its application in our study.

In McNemar's test, each method generates predicted labels for each sample in the dataset through cross-validation. To compare method A with method B using McNemar's test, one needs to derive the number of samples misclassified by A but not by B (denoted as n_{01}) and the number of samples misclassified by B but not by A (denoted as n_{10})). Subsequently, the test statistic $\frac{(|n_{01}-n_{10}|-1)^2}{n_{01}+n_{10}}$ is computed, which follows a chi-square (χ^2) distribution with one degree of freedom. This statistic quantifies the significance of the performance difference between the two methods, allowing us to make informed decisions about the superiority of one approach over the other in the context of sentiment analysis.

IV. RESULTS

Here, we report the performance of the PLMs and SOTA, and five FPLMs on the six datasets.

A. COMPARISON OF THE DOMAIN-INDEPENDENT PLMs

Table 4 presents the performances (in precision, recall, and F-score using percentages) of the PLMs in detecting sentiments over the GIT, ARC, SOC, and MAC datasets. The macro average for each metric is computed over all the sentiments presented in the last three columns of the table. For each dataset, we highlight the best and worst values for each metric across all the PLMs using blue and red colors, respectively. We also **boldface** the best value a PLM achieves for a metric within its category (e.g., TPLM, ZPLM, and FPLM). Similarly, Table 5 and 6 show the performances of the PLMs on the JIC and CRC datasets (that have two labels instead of three), respectively. In the following, we describe the results for each dataset.

1) GIT

Overall, the TPLMs achieve the best performance in detecting sentiments, with macro average F1-scores between 90.4% and 92.3%. The base ZPLMs perform significantly worse, with macro average F1-scores around 43.5% to 49.6% with an exception for the GPT models where GPT-4 achieves F1-score over 76.6%. ZPLMs show improved performances after fine-tuning them, reaching macro average F1-scores between 71.8% and 74.4%. The FPLMs score between the TPLMs and ZPLMs, with a macro F1-score of around 86%.

2) ARC

The PLMs display consistent performance patterns across the GIT and ARC datasets. For example, the TPLMs and base ZPLMs show the best and worst performances, respectively, while the FPLMs fit in between. Comparing the API review results to the GIT dataset, PLMs' performances are noticeably higher overall on the GIT dataset. For example, BERT - being the best performer among the TPLMs in the ARC dataset - achieves a macro F1 of 82.4% versus 91.5% on the GIT dataset. This substantial gap suggests the GIT dataset is cleaner and less noisy for sentiment detection with a more balanced distribution of sentiment labels.

3) SOC AND MAC

The FPLMs achieve the best performance in SOC and MAC datasets in contrast to the GIT and ARC datasets. While the TPLMs show the second-best performance in SOC, they achieve the worst performance in MAC. The GPT-4 achieves the best performance among the base and fine-tuned ZPLMs in both datasets.

4) JIC

Again, in this dataset with two labels, FPLMs perform best, followed by the TPLMs. However, the performance gap between these two PLM categories is negligible. For example, All-DistilRoBERTa - the best performing PLM from the FPLM category - achieves only 0.2% higher F1-score than RoBERTa, which is the best performer from the TPLM category. Surprisingly, the base and fine-tuned ZPLMs perform significantly better in this dataset, with the highest macro F1-scores of 93.1% and 93.3% achieved by the base and fine-tuned RoBERTa-mnli, respectively.

5) CRC

Similar to the JIC dataset, the CRC dataset has two labels (i.e., non-negative and negative) where the TPLMs perform the best, followed by the FPLMs. All the PLMs perform better in detecting non-negative sentiment than negative sentiment. This better performance on non-negative sentiment can be attributed to the higher sample number in this category, which contributes to 75% of the full data. Unlike the JIC dataset, both the base and fine-tuned ZPLMs (except the

TABLE 4. Performances of the PLMs in GIT, ARC, SOC, and MAC datasets.

	PLM Cat.	PLM Name	Positive			Negative			Neutral			Macro Average		
			p	r	$F1$	p	r	$F1$	p	r	$F1$	p	r	$F1$
GIT	TPLM	BERT	**94.3**	93.2	93.7	88.8	90.3	89.5	91.6	**91.1**	91.3	91.6	91.5	91.5
		XLNET	91.9	94.8	93.3	89.7	89.5	89.5	92.4	90.3	91.3	91.3	91.6	91.4
		RoBERTa	93.2	**95.8**	**94.5**	**89.9**	**91.3**	**90.5**	**93.5**	90.5	**91.9**	**92.2**	**92.5**	**92.3**
		ALBERT	92.8	92.2	92.4	89.0	87.5	88.2	90.0	**91.1**	90.5	90.6	90.3	90.4
	ZPLM	BART-mnli	55.5	77.3	64.6	46.4	93.6	62.0	58.4	2.0	3.9	53.4	57.6	43.5
		RoBERTa-mnli	61.8	77.2	68.6	46.9	94.5	62.7	73.6	9.9	17.4	60.8	60.5	49.6
		DeBERTa-mnli	57.6	**85.6**	68.9	48.8	95.0	64.5	51.0	1.0	2.0	52.5	60.6	45.1
		GPT-3.5	80.7	78.9	79.8	41.0	**98.0**	57.8	**97.3**	4.8	9.2	73.0	60.6	48.9
		GPT-4	**81.2**	81.1	**81.1**	**81.1**	65	**72.1**	72	**81.9**	**76.7**	**78.1**	**76**	**76.6**
	FPLM	all-MiniLM-L6-v2	**92.8**	91.8	92.3	83.3	80.8	82.0	81.8	84.3	83.0	85.9	85.6	85.8
		All-MPNET-base	92.1	**94.1**	**93.1**	86.6	82.1	84.3	84.3	**85.7**	**85.0**	87.7	87.3	87.5
		All-DistilRoBERTa	91.7	93.8	92.7	**87.2**	83.1	**85.1**	84.4	85.6	**85.0**	**87.8**	**87.5**	**87.6**
		GPT-3.5F	78.5	88.4	83.2	63.4	83.8	72.2	**86.2**	57.2	68.8	76.0	76.5	74.7
		LaBSE	91.6	93.8	92.7	81.6	**84.8**	83.1	85.7	81.5	83.6	86.3	86.7	86.5
	FT. ZPLM	BART-mnli	84.1	64.7	73.1	**69.4**	83.5	**75.7**	72.9	**73.5**	**73.2**	75.5	73.9	74.0
		RoBERTa-mnli	**84.5**	**70.0**	**76.4**	68.5	81.7	74.4	**73.7**	71.6	72.3	**75.6**	**74.4**	**74.4**
		DeBERTa-mnli	79.6	69.7	74.2	64.7	**85.8**	73.7	73.5	62.8	67.6	72.6	72.7	71.8
ARC	TPLM	BERT	**81.3**	**83.8**	**82.5**	**72.3**	**70.6**	**71.1**	**94.1**	**93.4**	**93.7**	**82.6**	**82.6**	**82.4**
		XLNET	71.2	82.8	76.1	62.3	63.4	61.6	93.3	87.8	90.4	75.6	78.0	76.0
		RoBERTa	77.9	82.1	79.4	71.5	68.8	68.7	93.4	91.5	92.3	80.9	80.8	80.1
		ALBERT	78.9	81.1	79.7	66.8	67.4	66.8	93.2	91.9	92.5	79.6	80.1	79.6
	ZPLM	BART-mnli	24.2	78.3	37.0	21.4	66.2	32.4	82.5	3.0	5.8	42.7	49.2	25.1
		RoBERTa-mnli	28.8	71.3	41.0	18.1	70.7	28.9	86.7	10.6	18.9	44.5	50.9	29.6
		DeBERTa-mnli	24.1	**81.0**	37.2	23.1	66.8	34.3	73.1	2.2	4.4	40.1	50.0	25.3
		GPT-3.5	44.1	50.1	47.2	14.7	**85.7**	25.1	**89.9**	14.7	25.3	49.6	50.2	32.5
		GPT-4	**48.9**	54.6	**51.6**	**31.7**	49.5	**38.6**	75.6	**89.7**	**82.0**	**53.6**	**69.6**	**60.3**
	FPLM	all-MiniLM-L6-v2	**61.1**	58.7	59.9	40.3	34.7	37.3	83.8	86.6	**85.2**	61.7	60.0	60.8
		All-MPNET-base	51.7	53.3	52.5	42.4	28.0	33.7	81.0	84.6	82.7	58.3	55.3	56.3
		All-DistilRoBERTa	61.0	57.5	59.2	**46.5**	29.0	35.7	81.9	**88.1**	84.9	**63.1**	58.2	59.9
		GPT-3.5F	41.3	62.4	49.7	25.6	**63.4**	36.5	**88.3**	53.6	66.7	51.7	59.8	51.0
		LaBSE	58.4	**63.6**	**60.9**	38.7	49.0	**43.2**	86.7	80.9	83.7	61.3	**64.5**	**62.6**
	Ft. ZPLM	BART-mnli	**42.8**	57.9	48.9	**28.9**	61.8	**39.3**	85.1	**60.6**	**70.8**	**52.3**	60.1	**53.0**
		RoBERTa-mnli	41.1	64.4	**50.1**	27.0	65.2	38.1	**88.6**	54.2	67.0	**52.3**	**61.2**	51.8
		DeBERTa-mnli	35.8	**65.8**	45.9	26.0	**65.6**	37.1	85.6	42.8	56.0	49.1	58.1	46.3
SOC	TPLM	BERT	67.0	50.4	55.6	72.7	70.6	71.4	91.6	**94.1**	92.8	77.1	71.7	73.3
		XLNET	57.2	**67.3**	59.3	69.4	**83.3**	75.3	**94.6**	88.3	91.2	73.7	**79.7**	75.3
		RoBERTa	**71.7**	54.2	**59.9**	**76.6**	79.4	**77.0**	92.8	93.9	**93.3**	**80.4**	75.9	**76.7**
		ALBERT	58.4	**48.5**	51.2	67.3	63.3	64.4	90.9	92.6	91.7	72.2	68.1	69.1
	ZPLM	BART-mnli	12.3	87.3	21.5	30.4	91.9	45.6	73.3	1.7	3.3	38.6	60.3	23.5
		RoBERTa-mnli	14.9	84.6	25.4	27.6	95.0	42.7	93.4	11.1	19.7	45.3	63.6	29.3
		DeBERTa-mnli	11.7	**93.1**	20.8	37.5	91.9	**53.3**	60.6	1.3	2.6	36.6	62.1	25.6
		GPT-3.5	24.0	78.6	36.7	23.7	**98.9**	38.3	**99.0**	27.5	43.0	48.9	68.3	39.3
		GPT-4	**46.9**	70.2	**56.2**	**81.4**	37.0	50.9	87.8	**90.1**	**89.0**	**72.0**	**65.8**	**65.4**
	FPLM	all-MiniLM-L6-v2	70.1	73.9	71.9	62.8	78.3	69.7	**96.6**	94.7	95.6	76.5	82.3	79.1
		All-MPNET-base	71.6	74.8	73.1	**65.7**	66.7	66.2	95.7	95.2	95.5	77.7	78.9	78.3
		All-DistilRoBERTa	**75.5**	74.8	**75.1**	64.6	76.8	**70.2**	96.5	**95.5**	**96.0**	**78.9**	**82.4**	**80.4**
		GPT-3.5F	26.2	**79.3**	39.4	44.9	**92.7**	60.5	96.0	57.5	71.9	55.7	76.5	57.3
		LaBSE	67.5	74.8	70.9	58.9	76.8	66.7	**96.6**	94.0	95.3	74.3	81.8	77.6
	FT. ZPLM	BART-mnli	23.4	75.4	35.2	36.9	82.5	50.8	91.6	49.0	63.0	50.6	69.0	49.7
		RoBERTa-mnli	**27.6**	**87.7**	**41.8**	**44.6**	**94.7**	**60.4**	**97.4**	**56.7**	**71.5**	**56.5**	**79.7**	**57.9**
		DeBERTa-mnli	13.8	84.2	23.6	39.8	84.4	53.6	81.7	20.7	30.9	45.1	63.1	36.0
MAC	TPLM	BERT	84.9	87.1	85.6	78.2	86.9	81.6	0.0	0.0	0.0	54.4	58.0	55.8
		XLNET	88.3	**92.9**	90.4	83.0	91.2	86.6	0.0	0.0	0.0	57.1	61.3	59.0
		RoBERTa	**88.9**	92.6	**90.5**	**87.1**	**92.3**	**89.2**	**2.9**	**4.0**	**3.3**	**59.6**	**63.0**	**61.0**
		ALBERT	85.1	87.9	86.1	77.8	86.9	81.5	0.0	0.0	0.0	54.3	58.3	55.9
	ZPLM	BART-mnli	89.4	95.5	92.3	89.4	95.0	92.0	15.0	4.0	6.2	64.6	64.8	63.5
		RoBERTa-mnli	88.2	96.6	92.1	89.1	86.5	87.7	9.2	6.0	7.2	62.2	63.0	62.4
		DeBERTa-mnli	89.8	**98.4**	93.9	**89.8**	93.1	91.3	15.0	4.0	6.2	64.9	65.2	63.8
		GPT-3.5	**97.0**	88.2	92.4	74.8	**98.4**	85.0	0.0	0.0	0.0	57.3	62.2	59.1
		GPT-4	96.2	96.2	**96.2**	87.6	**98.4**	**92.7**	**77.7**	**28.0**	**41.1**	**87.2**	**74.2**	**76.7**
	FPLM	all-MiniLM-L6-v2	94.6	94.6	94.6	89.6	92.3	90.9	76.2	64.0	69.6	86.8	83.6	85.0
		All-MPNET-base	**100.0**	**95.7**	**97.8**	92.4	93.8	93.1	61.3	**76.0**	67.9	84.6	**88.5**	86.3
		All-DistilRoBERTa	98.9	**95.7**	97.3	91.2	**96.2**	**93.6**	75.0	72.0	**73.5**	88.4	88.0	**88.1**
		GPT-3.5F	90.7	89.7	90.2	**92.6**	67.6	78.2	25.0	24.0	24.4	69.4	60.4	64.3
		LaBSE	97.2	94.1	95.6	88.7	**96.2**	92.3	**80.0**	64.0	71.1	**88.6**	84.7	86.3
	FT. ZPLM	BART-mnli	93.3	92.1	92.4	85.2	**95.4**	89.7	24.8	10.0	14.2	67.7	65.8	65.1
		RoBERTa-mnli	**97.4**	78.7	86.8	**95.6**	63.5	75.9	18.6	**76.0**	**29.8**	**70.5**	**72.7**	64.1
		DeBERTa-mnli	93.0	**92.9**	**92.6**	89.1	91.2	**89.8**	**27.2**	20.0	23.0	69.8	68.0	**67.3**

TABLE 5. Performance of the PLMs on the JIC dataset.

Data→		JIC								
Cat.	PLM	Positive			Negative			Macro Avg.		
		p	r	$F1$	p	r	$F1$	p	r	$F1$
TPLM	BERT	97.4	94.3	95.8	97.5	98.8	98.1	97.4	96.6	97.0
	XLNET	95.9	93.6	94.7	97.1	98.1	97.6	96.5	95.9	96.1
	RoBERTa	**97.6**	95.3	**96.4**	97.9	**98.9**	**98.4**	**97.8**	**97.1**	**97.4**
	ALBERT	96.1	**95.9**	95.9	**98.1**	98.2	98.2	97.1	97.0	97.0
ZPLM	BART-mnli	80.0	93.6	86.2	96.9	89.2	92.9	88.4	91.4	89.5
	RoBERTa-mnli	**85.0**	97.6	**90.8**	98.8	**92.0**	**95.3**	**91.9**	**94.8**	**93.1**
	DeBERTa-mnli	79.4	97.6	87.5	98.8	88.3	93.3	89.1	93.0	90.4
	GPT-3.5	79.9	93.4	86.1	**99.8**	84.0	91.2	89.9	88.7	88.7
	GPT-4	77.6	**98.2**	86.7	99.2	84.1	91.0	88.4	91.1	88.8
FPLM	all-MiniLM-L6-v2	95.2	95.9	95.5	98.1	97.8	98.0	96.7	96.8	96.7
	All-MPNET-base	95.3	97.6	96.4	98.9	97.8	98.3	97.1	97.7	97.4
	All-DistilRoBERTa	**95.9**	97.6	**96.8**	98.9	**98.1**	**98.5**	**97.4**	**97.8**	**97.6**
	GPT-3.5F	79.2	**99.7**	88.2	**99.8**	87.8	93.4	89.5	93.8	90.8
	LaBSE	94.5	95.5	95.0	97.9	97.5	97.7	96.2	96.5	96.4
FT. ZPLM	BART-mnli	78.8	98.1	87.2	99.1	87.5	92.8	88.9	92.8	90.0
	RoBERTa-mnli	84.7	**98.8**	**91.2**	**99.4**	91.8	**95.4**	**92.1**	**95.3**	**93.3**
	DeBERTa-mnli	**85.6**	96.9	90.8	98.5	**92.4**	95.3	**92.1**	94.6	93.0

TABLE 6. Performance of the PLMs on the CRC dataset.

Data→		CRC								
Cat.	PLM	Non-Negative			Negative			Macro Avg.		
		p	r	$F1$	p	r	$F1$	p	r	$F1$
TPLM	BERT	91.5	90.1	90.7	72.3	74.6	73.0	81.9	82.4	81.9
	XLNET	90.2	**92.7**	**91.3**	**77.1**	69.0	71.8	**83.6**	80.8	81.5
	RoBERTa	**92.4**	89.7	90.9	72.5	**77.5**	**74.9**	82.4	**83.6**	**82.9**
	ALBERT	89.5	89.6	89.4	69.8	67.8	67.9	79.6	78.7	78.7
ZPLM	BART-mnli	83.0	73.7	78.0	41.2	54.8	46.9	62.1	64.2	62.5
	RoBERTa-mnli	89.6	67.2	76.8	43.8	76.6	55.7	66.7	71.9	66.3
	DeBERTa-mnli	89.0	49.5	63.6	35.1	81.6	49.0	62.0	65.6	56.3
	GPT-3.5	**94.6**	51.2	66.5	39.5	**89.6**	54.8	67.1	70.4	60.7
	GPT-4	92.2	**81.2**	**86.4**	**59.3**	74.8	**66.2**	**75.5**	**78.0**	**76.3**
FPLM	all-MiniLM-L6-v2	86.2	93.3	89.6	72.9	54.8	62.6	79.5	74.0	76.1
	All-MPNET-base	85.3	**94.4**	89.7	75.2	51.0	60.8	80.3	72.7	75.2
	All-DistilRoBERTa	88.3	93.9	**91.0**	**77.3**	62.3	**69.0**	**82.8**	**78.1**	**80.0**
	GPT-3.5F	**97.5**	42.9	59.5	35.8	**96.0**	52.1	66.7	69.5	55.8
	LaBSE	86.2	85.4	85.8	57.2	58.8	58.0	71.7	72.1	71.9
FT. ZPLM	BART-mnli	89.3	60.7	72.1	40.2	78.1	52.9	64.7	69.4	62.5
	RoBERTa-mnli	**91.1**	61.1	72.8	**41.6**	**81.6**	**54.9**	**66.4**	**71.4**	**63.8**
	DeBERTa-mnli	88.4	**63.5**	**73.7**	40.8	74.7	52.6	64.6	69.1	63.2

GPT-4 model) show lower performances in this dataset, with F1-scores between 56.3% and 63.8%. Notably, the performance of GPT-3.5F is lower than the GPT-3.5 in this dataset, which is opposite compared to all other datasets where GPT-3.5F shows better performances than GPT-3.5.

TABLE 7. The best performing PLMs from each category.

	PLM Category			
Dataset	TPLM	ZPLM	FPLM	FT. ZPLM
GIT	**RoBERTa (92.3)**	GPT-4 (76.6)	All-DistilRoBERTa (87.6)	RoBERTa-mnli (74.4)
ARC	**BERT (82.4)**	GPT-4 (60.3)	LaBSE (62.6)	BART-mnli (53.0)
SOC	RoBERTa (76.7)	GPT-4 (65.4)	**All-DistilRoBERTa (80.4)**	RoBERTa-mnli (57.9)
MAC	RoBERTa (61.0)	GPT-4 (76.7)	**All-DistilRoBERTa (88.1)**	DeBERTa-mnli (67.3)
JIC	RoBERTa (97.4)	RoBERTa-mnli (93.1)	**All-DistilRoBERTa (97.6)**	RoBERTa-mnli (93.3)
CRC	**RoBERTa (82.6)**	GPT-4 (76.3)	All-DistilRoBERTa (80.0)	RoBERTa-mnli (63.8)

TABLE 8. McNemar's statistical test between the performance of RoBERTa and All-DistilRoBERTa for each dataset.

GIT	ARC	SOC	MAC	JIC	CRC
1300.945 **(0.000)**	2324.831 **(0.000)**	412.878 **(0.000)**	132.300 **(0.000)**	0.623 (0.430)	382.402 **(0.000)**

6) OVERALL SUMMARY

Table 7 presents the best-performing PLM from each category for each of the datasets. The best-performing PLM is decided based on the macro F1-score mentioned in parenthesis.

a: BEST PLM IN EACH DATASET (ROW-WISE)

For each dataset, the best-performing PLM's name and its F1-score are in boldface in the table. We observe that TPLMs are the top performers in the larger datasets, such as GIT and ARC. In contrast, the FPLMs show superior results in the small datasets, such as SOC and MAC. The lower performance of the TPLMs compared to the FPLMs can be attributed to the availability of less data to fine-tune the TPLMs, which hinders their learning of the patterns in the data.

Although the datasets JIC and CRC are small in size, both TPLMs and FPLMs show the best performances in those datasets. Achieving relatively better results in these small datasets by TPLM can be attributed to having two sentiment categories (i.e., positive and negative only) in contrast to the three sentiment categories in SOC and MAC. Having fewer sentiment categories also increases the proportion of the samples in each sentiment, which helps TPLMs in fine-tuning.

b: BEST PERFORMER IN EACH PLM CATEGORY (COLUMN-WISE)

By looking into the best performer in each PLM category, we see that among TPLMs, RoBERTa shows the best performance in five datasets. GPT-4 is the clear winner among all the base and fine-tuned ZPLMs, as it shows the best results in all datasets except JIC. Among the FPLMs, All-Distilroberta achieves the best results in all datasets except ARC.

7) STATISTICAL SIGNIFICANCE TEST

The empirical results show that TPLM RoBERTa outperforms other models on large datasets, while the FPLM All-DistilRoBERTa excels on small datasets. To assess the statistical significance of the performance differences between RoBERTa and All-DistilRoBERTa McNemar's test is applied. To do that, we formulate the following null and alternative hypotheses to determine the statistical significance of performance differences between the PLMs.

a: NULL HYPOTHESIS-1 (H_0^1)

There is no *significant* difference between the performance of RoBERTa and All-DistilRoBERTa.

b: ALTERNATIVE HYPOTHESIS-1 (H_A^1)

There exist *significant* differences between the performance of RoBERTa and All-DistilRoBERTa.

The McNemar's test results are presented in Table 8, with the p-values displayed in parentheses under the test statistic values. For all datasets except JIC, McNemar's test yields a p-value of 0.000, where $p < \alpha$ (the level of statistical significance that is set at 0.05). Consequently, the null hypothesis (H_0^1) is rejected for these datasets. Hence, the alternative hypothesis (H_a^1) holds true, i.e., there are significant performance differences exist between RoBERTa and All-DistilRoBERTa in the majority of cases. Based on our empirical observations and the results of McNemar's tests, we now derive the answer to the RQ1 as follows.

Ans. to RQ1: Performance of the PLMs vary across the datasets. The size of the datasets and the distributions of sentiment labels can impact the performance of the PLMs. For example, TPLMs show the best performances when a dataset is large with a balanced distribution of sentiments. Conversely, FPLMs show the best results when the dataset size is small. In addition, FPLMs show competitive performances compared to TPLMs in larger datasets. RoBERTa and All-DistilRoBERTa stand out as the best-performing PLMs from the TPLM and FPLM categories in most cases. GPT-4 shows the best results among the base and fine-tuned ZPLMs.

B. COMPARING THE BEST PERFORMING PLMs AGAINST THE STATE-OF-THE-ART AND DOMAIN-SPECIFIC TPLM

Here, we compare the performance of the best-performing PLMs found in RQ1, i.e, RoBERTa and All-DistilRoBERTa against the two domain-specific

TABLE 9. Performance comparison of `RoBERTa` and `All-DistilRoBERTa` against the domain-specific PLMs and state-of-the-art tools in GIT, ARC, SOC, and MAC datasets.

Dataset	Model and approach	Positive			Negative			Neutral			Macro Average		
		p	r	$F1$	p	r	$F1$		r	$F1$	p	r	$F1$
GIT	RoBERTa	93.2	95.8	94.5	89.9	**91.3**	90.5	**93.5**	90.5	91.9	92.2	92.5	92.3
	seBERT	**93.8**	**95.9**	**94.8**	**94.3**	87.2	**90.6**	91.3	**94.6**	**92.9**	**93.1**	**92.6**	**92.8**
	BERT_SE	85.0	87.8	86.4	86.7	67.3	75.8	79.6	90.2	84.6	83.8	81.8	82.3
	SentiCR	81.3	77.5	79.3	79.6	63.5	70.6	71.4	83.5	77.0	77.4	74.8	75.6
	SentiStrength-SE	84.1	76.5	80.1	80.5	73.2	76.7	75.5	84.7	79.8	80.0	78.1	78.9
	All-DistilRoBERTa	91.7	93.8	92.7	87.2	83.1	85.1	84.4	85.6	85.0	87.8	87.5	87.6
ARC	RoBERTa	77.9	82.1	79.4	71.5	**68.8**	68.7	**93.4**	91.5	92.3	80.9	80.8	80.1
	seBERT	**86.2**	**83.7**	**84.9**	**81.1**	68.6	**74.1**	93.2	96.2	**94.7**	**86.8**	**82.8**	**84.5**
	BERT_SE	75.2	35.9	48.6	70.0	7.0	12.8	76.7	**99.0**	86.4	74.0	47.3	49.3
	SentiCR	70.2	52.9	60.3	52.3	39.2	44.7	76.2	84.5	80.1	66.2	58.9	61.7
	SentiStrength-SE	65.3	37.6	47.7	44.9	30.3	36.2	77.6	90.8	83.6	62.6	52.9	55.8
	All-DistilRoBERTa	61.0	57.5	59.2	46.5	29.0	35.7	81.9	88.1	84.9	63.1	58.2	59.9
SOC	RoBERTa	71.7	54.2	59.9	76.6	**79.4**	77.0	92.8	93.9	93.3	80.4	75.9	76.7
	seBERT	71.1	61.1	64.0	**80.2**	75.5	**77.2**	92.6	94.3	93.4	**81.3**	77.0	78.2
	BERT_SE	0.0	0.0	0.0	0.0	0.0	0.0	79.0	**100.0**	88.4	26.4	33.3	29.4
	SentiCR	**81.5**	**77.7**	**79.5**	79.9	63.3	70.6	71.4	83.8	77.1	77.6	74.9	75.7
	SentiStrength-SE	26.7	17.3	20.9	43.1	13.8	21.0	81.7	93.2	87.1	50.5	41.4	43.0
	All-DistilRoBERTa	75.5	74.8	75.1	64.6	76.8	70.2	**96.5**	95.5	**96.0**	78.9	**82.4**	**80.4**
MAC	RoBERTa	88.9	92.6	90.5	87.1	92.3	89.2	2.9	4.0	3.3	59.6	63.0	61.0
	seBERT	91.3	92.6	91.6	84.2	89.2	85.8	6.7	8.0	7.2	60.8	63.2	61.5
	BERT_SE	78.7	68.4	73.2	58.3	80.7	67.7	0.0	0.0	0.0	45.7	49.7	46.9
	SentiCR	75.1	80.8	77.8	71.3	70.9	70.9	9.0	5.4	6.6	51.8	52.4	51.8
	SentiStrength-SE	76.3	80.2	78.2	**93.7**	28.8	44.1	7.0	30.0	11.4	59.0	46.3	44.5
	All-DistilRoBERTa	**98.9**	**95.7**	**97.3**	91.2	**96.2**	**93.6**	**75.0**	**72.0**	**73.5**	**88.4**	**88.0**	**88.1**

TABLE 10. Performance comparison of `RoBERTa` and `All-DistilRoBERTa` against the domain-specific PLMs and state-of-the-art tools in JIC dataset.

Dataset→	JIC								
Model and approach	Positive			Negative			Macro Average		
	p	r	$F1$	p	r	$F1$	p	r	$F1$
RoBERTa	97.6	95.3	96.4	97.9	98.9	98.4	97.8	97.1	97.4
seBERT	**98.2**	95.3	96.7	97.9	**99.2**	**98.5**	**98.1**	97.2	**97.6**
BERT_SE	90.0	77.5	83.3	90.3	96	93.1	90.1	86.8	88.2
SentiCR	88.2	80.8	84.3	86.5	89.7	88.1	87.3	85.3	86.2
SentiStrength-SE	93.7	93.7	93.7	97.4	97.3	97.3	95.5	95.5	95.5
All-DistilRoBERTa	95.9	**97.6**	**96.8**	**98.9**	98.1	**98.5**	97.4	**97.8**	**97.6**

TABLE 11. Performance comparison of `RoBERTa` and `All-DistilRoBERTa` against the domain-specific PLMs and state-of-the-art tools in CRC dataset.

Dataset→	CRC								
Model and approach	Non-Negative			Negative			Macro Average		
	p	r	$F1$	p	r	$F1$	p	r	$F1$
RoBERTa	**92.4**	89.7	90.9	72.5	**77.5**	74.4	82.4	**83.6**	82.6
seBERT	90.7	93.9	**92.1**	**82.2**	70.5	**74.6**	**86.4**	82.2	**83.4**
BERT_SE	75	**100**	85	0	0	0	37.5	50	42.8
SentiCR	80.7	77.1	78.8	54.6	61.7	57.8	67.6	69.4	68.3
SentiStrength-SE	81.1	94.6	87.3	57.6	25.4	35.2	69.4	60.0	61.3
All-DistilRoBERTa	88.3	93.9	91.0	77.3	62.3	69.0	82.8	78.1	80.0

TABLE 12. The results of the McNemar's statistical test of `RoBERTa`, `SentiCR`, and `All-DistilRoBERTa` against `seBERT` for each dataset.

Model	GIT	ARC	SOC	MAC	JIC	CRC
RoBERTa	0.165 (0.684)	2.548 (0.110)	4.513 (**0.034**)	6.017 (**0.014**)	4.678 (0.801)	0.006 (0.937)
SentiCR	199.605 (**0.000**)	252.111 (**0.000**)	61.633 (**0.000**)	12.675 (**0.000**)	58.017 (**0.000**)	50.575 (**0.000**)
All-DistilRoBERTa	1217.524 (**0.000**)	2461.564 (**0.000**)	443.508 (**0.000**)	114.244 (**0.000**)	4.810 (0.816)	382.402 (**0.000**)

TPLMs `seBERT` [62] and `BERT_SE` [63], and the two state-of-the-art tools `SentiCR` [19], `SentiStrength-SE` [16].

For the above-selected PLMs and tools, Table 9 presents the results for the GIT, ARC, SOC, and MAC datasets.

Tables 10 and 11 report the results for the JIC and CRC datasets, respectively. The best results are bold-faced for the metrics in each dataset.

With respect to RQ1, here we see a similar trend in the performances of the PLMs that depend on the types of PLMs and the size of the datasets. For example, in the datasets with three sentiment categories, the RoBERTa and seBERT perform best in the larger datasets, such as GIT and ARC, and All-DistilRoBERTa performs best in smaller datasets, such as SOC and MAC. In the datasets with two sentiment categories, the RoBERTa, seBERT, and All-DistilRoBERTa show better results compared to the other selected PLMs and tools here. For example, seBERT and All-DistilRoBERTa achieve the same macro F1 scores (97.6%) in JIC dataset. In CRC dataset, seBERT achieves the best macro F1 score of 83.4%, which is 1.8% and 3.4% higher than the RoBERTa and All-DistilRoBERTa.

We also observe that the domain-specific seBERT always performs better than the domain-independent RoBERTa. Despite BERT_SE being a domain-specific BERT model tailored for software engineering tasks, it exhibits the worst performance among the evaluated approaches. However, it is noteworthy that BERT_SE's performance improves when the dataset is larger and more balanced, as evidenced by its third-rank position based on macro F1-score on the GitHub dataset, which has a total of 7,122 data of balanced sentiment (28.32% positive, 29.3% negative and 42.2% neutral data). Conversely, when the dataset is imbalanced and small, such as in the cases of MAC (54.5% positive, 38.1% negative, and 7.3% neutral, totaling only 341 data) and CRC (24.9% negative and 75.1% non-negative, totaling 1,600 data), BERT_SE's performance deteriorates. The performance of SentiStrength-SE falls between that of SentiCR and BERT_SE.

Statistical Significance Test. After analyzing the results, we conduct McNemar's statistical test to test the significance of the performance difference of RoBERTa, All-DistilRoBERTa, and SentiCR against seBERT. We formulate the following null and alternative hypotheses for each dataset as follows.

Null Hypothesis-2 (H_0^2): There is no *significant* difference between the performances of the seBERT and $\mathcal{X}$.

Alternative Hypothesis-2 (H_a^2): There exist *significant* differences between the performances of the seBERT and $\mathcal{X}$.

In the above two hypotheses, $\mathcal{X} \in$ {RoBERTa, All-DistilRoBERTa, and SentiCR}.

The McNemar's test results are presented in Table 12 where the p-values are displayed in parentheses under the test statistic values. We see that the seBERT significantly outperforms the RoBERTa in SOC and MAC datasets. Again, the seBERT significantly outperforms the SentiCR in all datasets. The performance differences between the seBERT and All-DistilRoBERTa are significant in all datasets except JIC where they achieve the same macro F1 score.

Hence, we answer the research question RQ2 as follows:

Ans. to RQ2: The domain-specific seBERT model demonstrates superior performance compared to the PLMs in sentiment detection tasks within the software engineering domain, across the majority of the datasets evaluated. However, in smaller datasets with limited training samples, the All-DistilRoBERTa model exhibits better results compared to seBERT. The performance differences between seBERT and All-DistilRoBERTa are statistically significant in all datasets, with the exception of the JIC dataset. While seBERT outperforms RoBERTa, the significant difference is only observed in the SOC and MAC datasets. Notably, both seBERT and RoBERTa surpass the performance of state-of-the-art tools across all datasets. Furthermore, All-DistilRoBERTa also outperforms the state-of-the-art tools in all datasets, with the exception of the ARC dataset.

V. ERROR ANALYSIS

We conduct the error analysis to supplement the quantitative analysis and gain a deeper understanding of where and why the PLMs are still limited. To do that, we carry out the error analysis in two steps. First, we focus on the errors generated by the domain-independent PLMs. Then, we conduct seBERT-specific error analysis to identify in what cases it performs better than the domain-independent PLMs and vice versa. We select the TPLMs for error analysis, as they show the best results in the majority of the datasets. The steps are described as follows.

A. STEP-1: DOMAIN-INDEPENDENT TPLM

1) GOAL

Find out the dominant causes of the errors of the four domain-independent TPLMs (i.e., BERT, RoBERTa, XLNeT, and ALBERT) in predicting sentiments.

2) PROCESS

For each selected TPLM, we obtain its predicted sentiment labels for the fold-1 test set of each dataset used in the experiments for RQ1. We exclude the CRC dataset as it combines positive and neutral sentiments into a non-negative category, making it unsuitable for merging with other datasets.

Upon comparing the predicted results with ground truth labels, we find that all four selected PLMs provide incorrect predictions for 142 comments. Consequently, an in-depth qualitative analysis is performed on those 142 samples. Two raters (the first two authors of this paper) jointly evaluate the samples and categorize the causes of the errors. In the process of error analysis, they use the XAI tool SHAP [33].

3) FINDINGS

A total of seven error types are identified and presented in Table 13 that are decreasingly ordered based on their distributions.

a: GENERAL ERRORS

The most dominant cause of errors is general error, accounting for 23.9% of errors. These errors occur when models fail to detect language patterns or missing/misleading word and emoticon cues. For example, the negatively rated post "*yes i noticed that after i send this pr -.-*" mislabeled as neutral as the PLMs fail to detect the emoticon'-.-' cue properly in the post that a user usually uses to express negative sentiment.

To better understand the issue, we analyze the explanation of the SHAP tool for the PLM output presented in Figure 2. After analyzing the SHAPE interpretation, we find that the emoticon '-.-' is tokenized in three sequential characters -,., and -. Moreover, the technical word 'pr' (pull request) is unknown to the PLM and is tokenized in two characters. All these characters are used as features and mislead the PLM in predicting the 'LABEL_0', i.e., neutral as indicated by the deepest red color. Here, 'LABEL_1' and 'LABEL_2' represent positive and negative sentiments, respectively. At the bottom of the figure, the words/characters in red indicate those that positively contribute to predicting the selected label, whereas the blue color indicates the opposite.

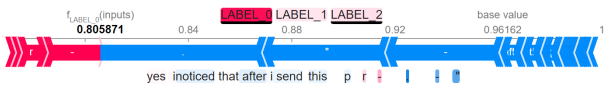

FIGURE 2. SHAP explanation of misinterpretation of emoticon.

b: LABELING ERROR

The second dominant cause is human error in labeling. This error type consists of 21.8% of all mistakes. For example, the item, "*So best keep max one nullable item per line.*" - is obviously a neutral comment but rated as positive in the ARC dataset. Such labeling errors by humans are also reported elsewhere [74]. It is notable that 67.67% of such labeling errors are located in only the ARC dataset.

c: EXPLICIT SENTIMENT

The PLMs misinterpret samples that do not contain explicit sentimental words, and such errors are liable for 19% of the total errors. For instance, the negatively rated post "*Also it doesn't specify the mode; read Wikipedia's Block Cipher Modes of Operation for concerns.*" - does not have any sentimental words that mislead the PLMs to interpret it as neutral.

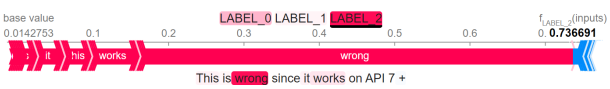

FIGURE 3. SHAP explanation for a prediction of a polar fact.

d: POLAR FACTS

The comments that report positive or negative facts, i.e., polar facts, are often mislabeled as sentimental by the PLMs due to the polarized word cues in those. For instance, the neutrally rated post "*This is wrong since it works on API 7+*" simply conveys a negative fact but is misinterpreted as a negative comment by the PLMs due to the word 'wrong' as evident in the SHAP's explanation in Figure 3. Polar facts in sentiment labeling are the fourth largest cause of misclassifications at 12.6%

TABLE 13. Distribution of error types in datasets.

Error Type	Datasets					# (%)
	ARC	GIT	JIC	MAC	SOC	
General error	14	14	-	-	6	34 (23.9)
Labeling error	18	5	2	-	6	31 (21.8)
Implicit Senti.	3	10	6	1	7	27 (19.0)
Polar fact	8	7	-	-	3	18 (12.6)
Subjectivity	6	2	2	-	8	18 (12.6)
Politeness	1	3		-	4	8 (5.6)
Figurative	1	3	2	-	-	6 (4.2)
Grand Total	51	44	12	1	34	142 (100%)

e: SUBJECTIVITY IN ANNOTATION

Subjectivity in sentiment labeling has the same proportion as the polar facts that causes 12.6% of errors. Sentiment analysis is inherently subjective as people perceive emotions differently. For example, the comment "*I need all subclasses to execute CODE_FRAGMENT, but I can't rely on them doing it explicitly.*" is annotated as positive, although the PLMs detect neutral sentiment in that.

f: POLITENESS

The presence of politeness phrases accounted for 5.6% of errors. For example, the neutrally rated post "*I appreciate your help.*" is misclassified as a positive comment by the PLMs due to the word "appreciate".

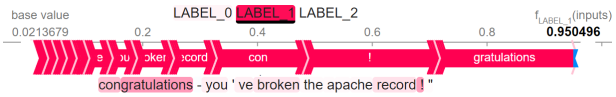

FIGURE 4. SHAP explanation of mislabeling a figurative comment.

g: FIGURATIVE EXPRESSIONS

Figurative, i.e., irony and sarcasm language, remains the least minor cause, accounting for 4.2% of the errors. For example, the negatively polarized post "*Congratulations - you've broken the Apache record!*" is misclassified as having positive sentiment due to the word 'congratulation' in it as confirmed by the SHAP's output in Figure 4. However, the word is used to express irony here.

B. seBERT-SPECIFIC ERROR ANALYSIS

1) GOAL

As we observe from RQ2, the domain-specific model seBERT outperforms RoBERTa, albeit by a minimal margin. Here, we aim to identify the reasons behind seBERT's superior performance compared to RoBERTa.

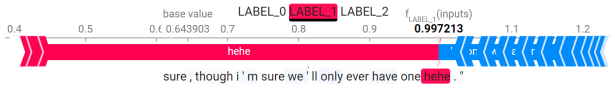

FIGURE 5. SHAP explanation of detecting internet slang word by seBERT.

FIGURE 6. SHAP explanation of misinterpretation of internet slang word by RoBERTa.

2) PROCESS

For the seBERT and RoBERTa, We obtain their predicted sentiment labels for the fold-1 test set of the GIT dataset used in the experiments for RQ1. We select the GIT dataset as it is the largest among the six datasets under consideration, as well as its balanced nature, consisting of three sentiment categories: positive, negative, and neutral. Upon reviewing the predicted results, we identify 51 instances out of the 1,424 test items in the fold-1 test set where seBERT provides correct predictions, while RoBERTa's predictions are wrong. To investigate the underlying causes of this discrepancy, two raters (the first two authors of this thesis) jointly evaluate these 51 samples with the help of SHAP tool.

3) FINDINGS

After analyzing the error instances, we find the following two key reasons for seBERT's superior performance over RoBERTa.

a: seBERT *CAN MORE EFFECTIVELY RECOGNIZE INTERNET SLANG WORDS*

. Software developers use frequently use internet slang and being domain-specific seBERT performs better in detecting those internet slang. For example, the comment "*Sure, though I'm sure we'll only ever have one hehe.?*" - with the Internet slang "hehe" that expresses amusement - is rated with positive sentiment by human raters. seBERT correctly identifies the slang and detects positive sentiment in it. In contrast, RoBERTa fails to detect the slang and erroneously rates the comment with neutral sentiment.

To gain deeper insight into the outputs of the models, we examine the explanation provided by the SHAP tool. In Figure 5, we can see that seBERT can detect the word "hehe" and apply proper weight to it to detect the correct sentiment of the comment. On the other hand, as seen in Figure 6, ttRoBERTa breaks down the word hehe (first "he" in red and last "he" in blue color) as it is an unknown word to it and fails to detect the sentiment of the comment correctly.

b: seBERT *CAN MORE ACCURATELY IDENTIFY DOMAIN-SPECIFIC EXPRESSIONS AND WORDS*

seBERT also performs better than RoBERTa in detecting domain-specific expressions and words in a sentence. For example, the comment "*quit spamming my notifications, please, kthxbye*" is rated with negative sentiment by human raters. seBERT can detect the sentiment of the comment correctly for two reasons: (i) it detects the domain-specific term "Spamming" correctly as seen in Figure 7, whereas, RoBERTa breaks down the word spamming as in two different words" "spa" and "mming", (ii) seBERT understand the domain-specific expression in the comments and applies proper weights on the words "Spamming" and "Please" and correctly detects the negative sentiment. Note the word "Please" usually expresses a non-negative sentiment. In contrast, RoBERTa applies improper weight on the word "Please" that leads it to detect neutral sentiment as seen in Figure 8.

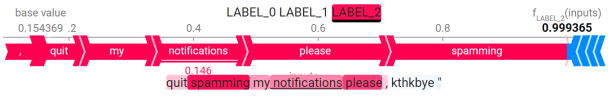

FIGURE 7. SHAP explanation of detecting domain-specific word by seBERT.

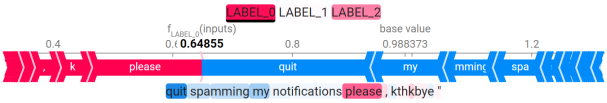

FIGURE 8. SHAP explanation of not identifying domain-specific word by RoBERTa.

Hence, we answer the research question RQ3 as follows:

Ans. to RQ3: General errors, labeling errors, explicit sentiments in comments, subjectivity in annotation, and figurative expressions are the dominant causes of the errors for the TPLMs in predicting the sentiments accurately. Pre-training seBERT with domain-specific corpus helps it to improve in detecting domain-specific expressions and words to avoid certain errors.

VI. DISCUSSION

This section describes the lessons learned from our experiments and provides an empirically derived guideline for selecting a PLM.

Lesson #1: PLMs can achieve reliable results. However, the dataset size and label distribution can impact the overall performance. The size and diversity of the training data play a critical role in the PLMs' ability to generalize across various sentiment expressions. In larger datasets with a balanced distribution of sentiment labels, TPLMs, such as RoBERTa, have proven to be reliable On the contrary, FPLMs tend to excel in smaller datasets. This can be attributed to their fine-tuning of specific tasks, which allows them to adapt better to limited data scenarios. Nonetheless, they also show competitive performance in larger datasets, indicating their versatility. GPT−4 can be used if a fine-tuning option is not available.

Guideline to select a PLM/tool. Based on this lesson, we derive a guideline presented in Figure 9 to help a user

TABLE 14. Comparative macro average F1 between GPT-4 and SentiStrength-SE.

Dataset	GPT-4	SentiStrength-SE
GIT	76.6	**78.9**
ARC	**60.3**	55.8
SOC	**65.4**	43.0
MAC	**76.7**	44.5
JIC	88.8	**95.5**
CRC	**76.3**	61.3

TABLE 15. McNemar's test between the performances of GPT-4 and SentiStrength-SE for each dataset.

GIT	ARC	SOC	MAC	JIC	CRC
6.273 (**0.012**)	194.026 (**0.000**)	2.403 (**0.011**)	108.245 (**0.000**)	658.002 (**0.000**)	23.602 (**0.000**)

in choosing a PLM based on different scenarios. It starts by asking if there is any labeled sentiment data available. If yes, a user can proceed to ask if there are at least $n \approx 300$ samples.[3] If yes, a TPLM, such as seBERT and RoBERTa, can be used by fine-tuning them. An FPLM, such as All-DistilRoBERTa, can be used if fewer labeled data are available.

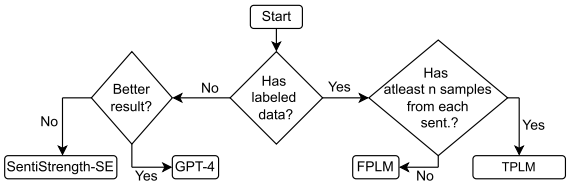

FIGURE 9. Empirically derived guideline to select a PLM/tool.

In scenarios where no labeled sentiment data is available for a specific domain, GPT-4 or the SentiStrength-SE can be two viable approaches, as neither requires domain-specific labeled data for sentiment analysis. Table 14 presents the macro-averaged F1 scores achieved by GPT-4 and SentiStrength-SE across all datasets. The table shows that GPT-4 exhibits strong performance in most datasets, except GIT and JIC. To ascertain whether the observed performance differences between GPT-4 and SentiStrength-SE are statistically significant, we conduct McNemar's test. The results of McNemar's test comparing the predictions of GPT-4 and SentiStrength-SE are reported in Table 15 where we see the performance differences are significant. Thus, if a user expects better results and is able to pay for the service, GPT-4 can be used. Otherwise, SentiStrength-SE is available at no cost.

Lesson #2: The quality of the gold standard impacts the classification performance. Novielli et al. [9] find that the absence of clear guidelines for annotation leads to noisy gold standards, thus resulting in unreliable model training and testing. Such a scenario is also evident in our study, as the PLMs show the best performances in GIT and JIC datasets that are consistently annotated using a theoretical framework in contrast to other datasets (e.g., ARC) that are annotated using ad hoc procedures. The error analysis also identifies that 58.06% of the total labeling errors originated from the ARC dataset, which can be attributed to the absence of a clear guideline to annotate data.

Lesson #3: XAI technique offers a deeper understanding of the causes of errors with increased confidence. We use the XAI tool SHAP [33] to better understand the errors that help us to better understand the causes of errors. For example, using SHAP, we find that unknown technical terms, e.g., "pr" and emoticons in Figure 2 can create unnecessary word/character features that mislead a PLM. The PLMs should be able to identify emoticons properly or have emoticons be preprocessed, such as replacing them with tokens related to their meaning (e.g., "happy", "sad") [75]. Moreover, the other errors are identified with increased confidence with the help of SHAP as shown in Figure 3 and 4.

Lesson #4: Curating datasets with figurative and politeness expressions and using them to fine-tune the PLMs can improve their performances. We find that fewer number comments with figurative and politeness expressions in the training data is one of the main causes for the PLMs to detect those inaccurately. One possible solution to that is to curate a larger and more diverse set of comments that include figurative language and politeness. Mishra and Chatterjee [74] also suggested the same to improve emotion detection from figurative language in SE.

VII. RELATED WORK

In this section, we discuss the related work grouped into four categories: (i) benchmarking domain-independent SA tools in SE, (ii) comparing a newly developed SE domain-specific tool against existing ones, (iii) benchmarking SE domain-specific SA tools, and (iv) comparing PLMs against existing SA tools.

A. BENCHMARKING DOMAIN-INDEPENDENT SA TOOLS IN SE

Jongeling et al. [76] compared four *domain independent* tools, such as, NLTK [77], Stanford NLP [78], SentiStrength [15], and Alchemy [79] for SA in SE text. They found that these tools showed low performances in predicting sentiment labels and suggested a requirement for SE domain-specific SA tools.

B. COMPARING A NEWLY DEVELOPED SE DOMAIN-SPECIFIC TOOL AGAINST EXISTING ONES

Islam and Zibran developed the first SE domain-specific SA tool, SentiStrength-SE [80], and compared that against SentiStrength. Ahmed et al. [19] developed the SE domain-specific tool SentiCR and evaluated that again seven *domain independent* sentiment detection techniques (e.g., NLTK, SentiStrength). Calefato et al. [18] developed another SE domain-specific tool Senti4SD and compared its performance against SentiStrength and

[3]The value of n is carefully chosen based on our empirical evaluation results, where TPLMs show better results in the smaller dataset JIC with 290 positive samples.

SentiStrength-SE. Later, Chen et al. [81] introduced the tool SEntiMoji and compared the performance of it against Senti4SD, SentiCR, SentiStrength-SE. Their experimental results showed that SEntiMoji significantly outperformed the existing SA tools in SE.

C. BENCHMARKING SE DOMAIN-SPECIFIC SA TOOLS

Islam and Zibran [82] presented the first benchmarking study by evaluating three SE domain-specific SA tools, i.e., EmoTxt [10], Senti4SD, and SentiStrength-SE. Their study found that the individual tools exhibited their best performance on the dataset they were originally tested at the time of their release. Subsequently, Novielli et al. [8] compared four tools, i.e., Senti4SD, SentiStrength-SE, SentiCR, and SentiStrength on four SE domain-specific datasets. They found that Senti4SD achieved the highest macro-averaged F1-score for the Stack Overflow dataset, while SentiCR was the highest for the other three datasets.

D. COMPARING PLMs AGAINST EXISTING SA TOOLS

Zhang et al. [26] investigated the performance of four PLMs, namely BERT, RoBERTa, XLNet, and ALBERT, against five non-PLM tools that include three SE domain-specific tools - Senti4SD, SentiStrength-SE, and SentiCR, and two general-purpose tools - Stanford CoreNLP, and SentiStrength. Their experimental results showed that the best-performing PLM outperformed the best-performing non-PLM tool by 6.5% to 35.6% in the macro and micro-averaged F1 scores.

In the second study, Zhang et al. [84] compared the performance of the aforementioned PLMs and DistilBERT against Llama 2-Chat [85], Vicuna [86], and WizardLM [87] what they termed as big large language models (bLLMs). They found that with limited labeled data and pronounced class imbalance, prompting bLLMs is a more effective strategy, outperforming fine-tuning TPLMs. *However, our study finds in RQ1 that FPLMs can perform better than the GPT-based tools (bLLMs in our study) and TPLMs in a dataset with limited label data with class imbalance*. We find this result due to our diverse selection of models, which includes TPLMs, FPLMs, and bLLMs.

In another study, Uddin et al. [83] developed an ensemble tool called Sentisead where five stand-alone rule-based and shallow learning SE-specific tools' (such as Senti4SD, SentiCR, SentistrengthSE, Opiner, and POME) predictions are passed to RoBERTa with the Bag of Words (BoW) features to get the final prediction of each provided text. They found only a 0.5% increase over a stand-alone RoBERTa model over six domain-specific datasets. However, this study, being heavily focused on overall results across all six datasets, did not analyze the impacts of a dataset size and class imbalance on the performance of a model. Thus, unlike our study, this study could not guide what type of model should be chosen based on the dataset size and proportion of samples in each sentient label.

Delta Between the Study of Our Work and the Studies of Zhang et al. and Uddin et al.: As the studies by Zhang et al. [26], [84] and Uddin et al. [83] are closely related to our study, we provide the comparison between these three studies in Table 16 in model selection, robust experiment design, statistical analysis, and error analysis.

In our study, the selection of the models and tools are more diverse than the other three studies. For example, for the first time, we empirically evaluate the recent advanced techniques, GPT-4, GPT-3.5, and FPLMs with GPT3.5F to assess their effectiveness for SA in SE against four domain-independent and two domain-specific TPLMs. Further, we employ a semi-supervised fine-tuning technique, namely *self-training*, to fine-tune the ZPLMs without any labeled data. Moreover, we conduct an in-depth qualitative error analysis with the support of an XAI technique, namely SHAP, to find the areas needing improvement for SA in SE. Finally, from our in-depth quantitative and qualitative analyses, we have outlined the lessons learned and derived a guideline to select the suitable tool based on different scenarios for SA in SE, which are more accurate than the other studies.

VIII. THREAT TO VALIDITY

A. CONSTRUCT VALIDITY

We use three metrics: precision, recall, F1-score, and their macro averages to evaluate the classification performances of the PLMs. All those metrics have been commonly used in similar studies in SE [8], [9], [19], [26], [80], [81].

The categorizations of the selected PLMs into TPLM, FPLM, and ZPLM can be criticized. Note that those categories are defined and used by the machine learning community and the developers of the PLMs[4,5]. In addition, BERT and RoBERTa are categorized as fine-tuned models (similar to our TPLM) and GPT 3.5 and GPT 4 as zero-shot model elsewhere [88]. In our study, GPT-3.5 and GPT-4 are used as zero-shot models. Later, we have fine-tuned the GPT-3.5 with the few-shot technique and named as GPT-3.5F. In addition, the category *fine-tuned ZPLM* can be questioned as the base ZPLMs are fine-tuned with training data. However, unlike the fine-tuning process of TPLM and FPLM, no label data is required to fine-tune the base ZPLM. Thus, those fine-tuned ZPLMs are considered zero-shot PLMs in our study.

B. INTERNAL VALIDITY

The threats to internal validity revolve around internal factors, particularly concerning hyperparameters utilized to fine-tune the PLMs on SA datasets. The hyperparameters, such as the learning rate, number of epochs, and batch size, have

[4]https://huggingface.co/models?pipeline_tag=zero-shot-classification&sort=trending

[5]https://huggingface.co/SetFit

TABLE 16. Delta between our study and the studies of Zhang et al. and Uddin et al.

Category	Criteria	Studies of Zhang et al.		Uddin et al. [83]	Our study
		1st [26]	2nd [84]		
Model selection	GPT-based models, e.g, `GPT-4`	No	No	No	**Yes**
	DS TPLMs, e.g., `seBERT`	No	No	No	**Yes**
	DI TPLMs, e.g., `RoERTa`	**Yes**	**Yes**	**Yes**	**Yes**
	FPLMs, e.g, `LaBSE`	No	No	No	**Yes**
	BERT-based ZPLMs, e.g., `DeBERTa-mnli`	No	No	No	**Yes**
	Self-training to fine-tune ZPLM	No	No	No	**Yes**
	LLAMA-based models	No	**Yes**	No	No
	State-of-the-art tools, e.g., `SentiCR`	**Yes**	No	**Yes**	**Yes**
Robust experiment	K-fold cross-validation	No	No	**Yes**	**Yes**
Statistical analysis	Significance test of results	No	**Yes**	**Yes**	**Yes**
Error analysis	XAI-based error analysis with `SHAP`	No	No	No	**Yes**

DS= Domain-specific, DI=Domain independent

the potential to impact model performance, although the values employed in this study align with those commonly used in other studies [26]. The reason why we have not used a validation set in our experiment is that the sizes of the several datasets are so small that separating them into three sets would negatively affect the training and testing phases, especially for the small datasets. For example, the sizes of the datasets (except GitHub Pull Request and Commit Comments and API Reviews Comments) are between 341 to 1,600, and splitting them to 60% training, 20% validation, and 20% testing would leave a few samples for training that could cause underfitting. Although we could use validation sets for the larger datasets, that would result in an unfair comparison of the tools' performance in smaller and larger datasets. We used the hyperparameter settings that are recommended/used popularly elsewhere [26]. Like our study, many scientific articles on machine learning and deep learning used recommended hyperparameter settings and did not use any validation sets [18], [26], [88], [89], [90], [91] to conduct their experiments.

However, uncontrolled variations might still exist stemming from hyperparameters. To address potential variations caused by shuffling training data and stochastic elements within the optimization algorithm, we used a specific strategy. Initially, we partitioned each dataset into ten folds, with each fold comprising 80% training data and 20% test data. Subsequently, these ten folds were individually employed to train and test each PLM belonging to both the TPLM and ZPLM categories. This method allowed us to maintain consistency in the training and evaluation processes across different models and datasets, reducing the impact of randomness.

The prompts and the default temperatures used to measure the performances of the `GPT` models can impact their performances. Given the straightforward nature of sentiment detection, our objective was to formulate a simple prompt that includes the dataset name with the task description and the format of outputs. In the used prompt, the names of the datasets, such as Mobile App Review Comments and JIRA issue comments, provide useful information about the source of the dataset. Moreover, it includes both the task description (You classify the sentiment) and output format (sentiment_types, i.e., positive, negative, and neutral). Despite the provided format of the output, the `GPT` models were explaining their classification results. Such an explanation came with a variable number of sentences and made it harder to parse the assigned sentiment categories. After adding the instruction "*No explanation is needed for the output*", it stopped adding any explanation. The inclusion of the dataset name and adding instructions not to explain the output in the prompt are also used elsewhere [88].

C. EXTERNAL VALIDITY

Threats to external validity are related to the generalizability of the research and experiments. The benchmark datasets were collected from widely-used software development platforms - Stack Overflow for technical Q&A, Jira for issue tracking, and GitHub for collaborative coding with version control. Each platform represents a different common task in the software development workflow. Moreover, given the combined size and diversity of tasks represented in the datasets, they represent real communication patterns among software developers. Overall, the benchmark datasets can reasonably represent how software developers communicate. Moreover, to compute the performances of the PLMs, we have applied cross-validation that ensures more diverse data is included in the performance computation process. We have measured the performances of the PLMs on unified data to test the generalizability of the models. Such designs in the experiments reduce the threat of external validity.

D. RELIABILITY

The datasets used in this study are publicly available. Moreover, all the PLMs except the `GPT` and the state-of-the-art tools are freely available. To ensure reproducibility, this study's experimental settings and scripts have been made publicly available [34]. Therefore, it should be possible to replicate this study.

IX. CONCLUSION

This study presents the most comprehensive examinations to date on the use of Pretrained Language Models (PLMs) for sentiment analysis in software engineering. By rigorously analyzing the performance of Zero-shot, Few-shot, and

Fine-tuned Transformer models across six diverse datasets, this work significantly advances our understanding of the strengths and limitations of these techniques.

The results demonstrate that the choice of optimal PLM depends on key dataset characteristics. Fine-tuned general-purpose models, such as `seBERT` and `RoBERTa` excel when trained on large, balanced datasets. In contrast, few-shot models, such as `All-DistilRoBERTa` are better suited for smaller datasets where they can effectively learn from limited examples. Notably, few-shot PLMs achieve competitive results even on larger datasets, highlighting their versatility.

Complementing the quantitative evaluation, a qualitative error analysis using the SHAP technique uncovers areas for further improvement in sentiment analysis for software engineering. The lessons derived from this study, such as the importance of dataset size, label distribution, and the role of explainable AI techniques, provide valuable insights to guide future research endeavors.

In summary, this work makes significant strides in harnessing the power of advanced language models for sentiment analysis in the software engineering domain. The comprehensive quantitative and qualitative findings, along with the empirically derived guidelines for model selection, serve as a foundation for researchers and practitioners to effectively leverage these techniques and drive further advancements in this field. In the future, we plan to expand our experiment with open-source large language models, such as `Llama-2` and `Vicunia`.

REFERENCES

[1] D. Graziotin, X. Wang, and P. Abrahamsson, "Are happy developers more productive? The correlation of affective states of software developers and their self-assessed productivity," in *Proc. Int. Conf. Product-Focused Softw. Process Improvement*, 2013, pp. 50–64.

[2] M. Ortu, B. Adams, G. Destefanis, P. Tourani, M. Marchesi, and R. Tonelli, "Are bullies more productive? Empirical study of affectiveness vs. Issue fixing time," in *Proc. IEEE/ACM 12th Work. Conf. Mining Softw. Repositories*, May 2015, pp. 303–313.

[3] T. Lesiuk, "The effect of music listening on work performance," *Psychol. Music*, vol. 33, no. 2, pp. 173–191, Apr. 2005.

[4] M. Mantyla, B. Adams, G. Destefanis, D. Graziotin, and M. Ortu, "Mining Valence, Arousal, and Dominance–possibilities for detecting burnout and productivity?" in *Proc. IEEE/ACM 13th Work. Conf. Mining Softw. Repositories (MSR)*, May 2016, pp. 247–258.

[5] N. Cassee, F. Zampetti, N. Novielli, A. Serebrenik, and M. Di Penta, "Self-admitted technical debt and comments' polarity: An empirical study," *Empirical Softw. Eng.*, vol. 27, no. 6, pp. 1–20, Nov. 2022.

[6] E. Guzman, D. Azócar, and Y. Li, "Sentiment analysis of commit comments in GitHub: An empirical study," in *Proc. 11th Work. Conf. Mining Softw. Repositories*, May 2014, pp. 352–355.

[7] M. R. Islam and M. F. Zibran, "Towards understanding and exploiting developers' emotional variations in software engineering," in *Proc. IEEE 14th Int. Conf. Softw. Eng. Res., Manage. Appl. (SERA)*, Jun. 2016, pp. 185–192.

[8] N. Novielli, D. Girardi, and F. Lanubile, "A benchmark study on sentiment analysis for software engineering research," in *Proc. IEEE/ACM 15th Int. Conf. Mining Softw. Repositories (MSR)*, May 2018, pp. 364–375.

[9] N. Novielli, F. Calefato, D. Dongiovanni, D. Girardi, and F. Lanubile, "Can we use SE-specific sentiment analysis tools in a cross-platform setting?" in *Proc. IEEE/ACM 17th Int. Conf. Mining Softw. Repositories (MSR)*, May 2020, pp. 158–168.

[10] F. Calefato, F. Lanubile, and N. Novielli, "EmoTxt: A toolkit for emotion recognition from text," in *Proc. 7th Int. Conf. Affect. Comput. Intell. Interact. Workshops Demos (ACIIW)*, Oct. 2017, pp. 79–80.

[11] B. Lin, F. Zampetti, G. Bavota, M. Di Penta, M. Lanza, and R. Oliveto, "Sentiment analysis for software engineering: How far can we go?" in *Proc. IEEE/ACM 40th Int. Conf. Softw. Eng. (ICSE)*, May 2018, pp. 94–104.

[12] B. Lin, N. Cassee, A. Serebrenik, G. Bavota, N. Novielli, and M. Lanza, "Opinion mining for software development: A systematic literature review," *ACM Trans. Softw. Eng. Methodol.*, vol. 31, no. 3, pp. 1–41, Jul. 2022.

[13] N. Novielli, F. Calefato, F. Lanubile, and A. Serebrenik, "Assessment of off-the-shelf SE-specific sentiment analysis tools: An extended replication study," *Empirical Softw. Eng.*, vol. 26, no. 4, pp. 1–28, Jul. 2021.

[14] P. Tourani, Y. Jiang, and B. Adams, "Monitoring sentiment in open source mailing lists: Exploratory study on the apache ecosystem," in *Proc. 24th Annu. Int. Conf. Comput. Sci. Softw. Eng.*, 2014, pp. 34–44.

[15] M. Thelwall, K. Buckley, and G. Paltoglou, "Sentiment strength detection for the social Web," *J. Amer. Soc. Inf. Sci. Technol.*, vol. 63, no. 1, pp. 163–173, Jan. 2012.

[16] M. R. Islam and M. F. Zibran, "SentiStrength-SE: Exploiting domain specificity for improved sentiment analysis in software engineering text," *J. Syst. Softw.*, vol. 145, pp. 125–146, Nov. 2018.

[17] R. Jongeling, S. Datta, and A. Serebrenik, "Choosing your weapons: On sentiment analysis tools for software engineering research," in *Proc. IEEE Int. Conf. Softw. Maintenance Evol. (ICSME)*, Sep. 2015, pp. 531–535.

[18] F. Calefato, F. Lanubile, F. Maiorano, and N. Novielli, "Sentiment polarity detection for software development," *Empirical Softw. Eng.*, vol. 23, no. 3, pp. 1352–1382, Jun. 2018.

[19] T. Ahmed, A. Bosu, A. Iqbal, and S. Rahimi, "SentiCR: A customized sentiment analysis tool for code review interactions," in *Proc. 32nd IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE)*, Oct. 2017, pp. 106–111.

[20] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. North Amer. Chapter Assoc. Comput. Linguistics*, 2019, pp. 4171–4186.

[21] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized BERT pretraining approach," 2019, *arXiv:1907.11692*.

[22] S. Zhou, B. Shen, and H. Zhong, "Lancer: Your code tell me what you need," in *Proc. 34th IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE)*, Nov. 2019, pp. 1202–1205.

[23] Z. Yu, R. Cao, Q. Tang, S. Nie, J. Huang, and S. Wu, "Order matters: Semantic-aware neural networks for binary code similarity detection," *Proc. AAAI Conf. Artif. Intell.*, vol. 34, no. 1, pp. 1145–1152, Apr. 2020.

[24] S. Das, N. Deb, A. Cortesi, and N. Chaki, "Zero-shot learning for named entity recognition in software specification documents," in *Proc. IEEE 31st Int. Requirements Eng. Conf. (RE)*, Sep. 2023, pp. 100–110.

[25] G. Colavito, F. Lanubile, and N. Novielli, "Few-shot learning for issue report classification," in *Proc. IEEE/ACM 2nd Int. Workshop Natural Language-Based Softw. Eng. (NLBSE)*, May 2023, pp. 16–19.

[26] T. Zhang, B. Xu, F. Thung, S. A. Haryono, D. Lo, and L. Jiang, "Sentiment analysis for software engineering: How far can pre-trained transformer models go?" in *Proc. IEEE Int. Conf. Softw. Maintenance Evol. (ICSME)*, Sep. 2020, pp. 70–80.

[27] P.-H. Chi, P.-H. Chung, T.-H. Wu, C.-C. Hsieh, Y.-H. Chen, S.-W. Li, and H.-Y. Lee, "Audio AlBERT: A lite BERT for self-supervised learning of audio representation," in *Proc. IEEE Spoken Lang. Technol. Workshop (SLT)*, Jan. 2021, pp. 1–24.

[28] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. Le, "XLNet: Generalized autoregressive pretraining for language understanding," in *Proc. Neural Inf. Process. Syst.*, 2019, pp. 5753–5763.

[29] T. Brown et al., "Language models are few-shot learners," in *Proc. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 1877–1901.

[30] J. Ye, X. Chen, N. Xu, C. Zu, Z. Shao, S. Liu, Y. Cui, Z. Zhou, C. Gong, Y. Shen, J. Zhou, S. Chen, T. Gui, Q. Zhang, and X. Huang, "A comprehensive capability analysis of GPT-3 and GPT-3.5 series models," 2023, *arXiv:2303.10420*.

[31] Y. Xian, C. H. Lampert, B. Schiele, and Z. Akata, "Zero-shot learning—A comprehensive evaluation of the good, the bad and the ugly," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 9, pp. 2251–2265, Sep. 2019.

[32] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 4080–4090.

[33] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 4768–4777.

[34] (2024). *Replication Package*. [Online]. Available: https://figshare.com/s/f755c96cf9ddca724188

[35] T. Zhang, C. Xia, C.-T. Lu, and P. Yu, "MZET: Memory augmented zero-shot fine-grained named entity typing," in *Proc. 28th Int. Conf. Comput. Linguistics*, 2020, pp. 171–180.

[36] O. Levy, M. Seo, E. Choi, and L. Zettlemoyer, "Zero-shot relation extraction via reading comprehension," 2017, *arXiv:1706.04115*.

[37] P. Kumar Pushp and M. Mayank Srivastava, "Train once, test anywhere: Zero-shot learning for text classification," 2017, *arXiv:1712.05972*.

[38] S. Kumar and P. Talukdar, "NILE: Natural language inference with faithful natural language explanations," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, 2020, pp. 9560–9572.

[39] A. Williams, N. Nangia, and S. Bowman, "A broad-coverage challenge corpus for sentence understanding through inference," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, 2018, pp. 1112–1122.

[40] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in *Proc. Conf. Empirical Methods Natural Language Process. 9th Int. Joint Conf. Natural Lang. Process. (EMNLP-IJCNLP)*, 2019, pp. 3982–3992.

[41] (2023). *The SBERT Framework*. [Online]. Available: https://sbert.net/

[42] F. Feng, Y. Yang, D. Cer, N. Arivazhagan, and W. Wang, "Language-agnostic BERT sentence embedding," in *Proc. 60th Annu. Meeting Assoc. Comput. Linguistics*, 2022, pp. 878–891.

[43] K. Song, X. Tan, T. Qin, J. Lu, and T. Liu, "MpNET: Masked and permuted pre-training for language understanding," in *Proc. NIPS*, Apr. 2020, pp. 16857–16867.

[44] (2023). *All-MPNET-Base*. [Online]. Available: https://huggingface.co/sentence-transformers/all-mpnet-base-v2

[45] L. Tunstall, N. Reimers, U. Jo, L. Bates, D. Korat, M. Wasserblat, and O. Pereg, "Efficient few-shot learning with sentence transformers," in *Proc. Int. Conf. Empirical Methods Natural Lang. Process.*, 2022, pp. 4527–4540.

[46] Y. Xian, S. Sharma, B. Schiele, and Z. Akata, "F-VAEGAN-D2: A feature generating framework for any-shot learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 10267–10276.

[47] B. Romera-Paredes and P. Torr, "An embarrassingly simple approach to zero-shot learning," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 2152–2161.

[48] (2023). *Bart Large MNLI*. [Online]. Available: https://huggingface.co/facebook/bart-large-mnli

[49] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, 2020, pp. 7871–7880.

[50] (2023). *RoBERTa Large MNLI*. [Online]. Available: https://huggingface.co/FacebookAI/roberta-large-mnli

[51] (2023). *Deberta-Large-MNLI-Zero Cls*. [Online]. Available: https://huggingface.co/Narsil/deberta-large-mnli-zero-cls

[52] P. He, X. Liu, J. Gao, and W. Chen, "Deberta: Decoding-enhanced BERT with disentangled attention," in *Proc. Int. Conf. Learn. Represent.*, 2021, pp. 1–20.

[53] (2024). *GPT-4*. [Online]. Available: https://platform.openai.com/docs/models/gpt-4-turbo-and-gpt-4

[54] (2023). *Text Davinci 003*. [Online]. Available: https://community.openai.com/t/text-davinci-003-deprecated/582617

[55] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt, "Measuring massive multitask language understanding," 2021, *arXiv:2009.03300*.

[56] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou, "MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers," 2020, *arXiv:2002.10957*.

[57] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proc. 37th Int. Conf. Mach. Learn.*, vol. 119, 2020, pp. 1597–1607.

[58] (2023). *All Distilroberta V1*. [Online]. Available: https://huggingface.co/sentence-transformers/all-distilroberta-v1

[59] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter," 2019, *arXiv:1910.01108*.

[60] K. Heffernan, O. Çelebi, and H. Schwenk, "Bitext mining using distilled sentence representations for low-resource languages," in *Proc. Findings Assoc. Comput. Linguistics, EMNLP*, 2022, pp. 2101–2112.

[61] T. Pires, E. Schlinger, and D. Garrette, "How multilingual is multilingual BERT?" in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 4996–5001.

[62] J. von der Mosel, A. Trautsch, and S. Herbold, "On the validity of pre-trained transformers for natural language processing in the software engineering domain," *IEEE Trans. Softw. Eng.*, vol. 49, no. 4, pp. 1487–1507, Apr. 2023.

[63] E. Maria De Bortoli Fávero and D. Casanova, "BERT_SE: A pre-trained language representation model for software engineering," 2021, *arXiv:2112.00699*.

[64] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," 2020, *arXiv:1910.13461*.

[65] (2024). *GPT-3.5*. [Online]. Available: https://platform.openai.com/docs/models/gpt-4-turbo-and-gpt-41

[66] (2024). *All-MiniLM-L6-v2*. [Online]. Available: https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

[67] G. Uddin and F. Khomh, "Automatic mining of opinions expressed about APIS in stack overflow," *IEEE Trans. Softw. Eng.*, vol. 47, no. 3, pp. 522–559, Mar. 2021.

[68] L. Villarroel, G. Bavota, B. Russo, R. Oliveto, and M. Di Penta, "Release planning of mobile apps based on user reviews," in *Proc. IEEE/ACM 38th Int. Conf. Softw. Eng. (ICSE)*, May 2016, pp. 14–24.

[69] M. Ortu, A. Murgia, G. Destefanis, P. Tourani, R. Tonelli, M. Marchesi, and B. Adams, "The emotional side of software developers in JIRA," in *Proc. IEEE/ACM 13th Work. Conf. Mining Softw. Repositories (MSR)*, May 2016, pp. 480–483.

[70] D.-C. Li, Y.-H. Fang, and Y. M. F. Fang, "The data complexity index to construct an efficient cross-validation method," *Decis. Support Syst.*, vol. 50, no. 1, pp. 93–102, Dec. 2010.

[71] (2024). *StratifiedShuffleSplit*. [Online]. Available: hhttps://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedShuffleSplit.html

[72] A. Gera, A. Halfon, E. Shnarch, Y. Perlitz, L. Ein-Dor, and N. Slonim, "Zero-shot text classification with self-training," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2022, pp. 1107–1119.

[73] T. G. Dietterich, "Approximate statistical tests for comparing supervised classification learning algorithms," *Neural Comput.*, vol. 10, no. 7, pp. 1895–1923, Oct. 1998.

[74] S. Mishra and P. Chatterjee, "Exploring ChatGPT for toxicity detection in GitHub," in *Proc. ACM/IEEE 44th Int. Conf. Softw. Eng. New Ideas Emerg. Results*, Apr. 2024, pp. 6–10.

[75] P. Delobelle and B. Berendt, "Time to take emoji seriously: They vastly improve casual conversational models," 2019, *arXiv:1910.13793*.

[76] R. Jongeling, P. Sarkar, S. Datta, and A. Serebrenik, "On negative results when using sentiment analysis tools for software engineering research," *Empirical Softw. Eng.*, vol. 22, no. 5, pp. 2543–2584, Oct. 2017.

[77] (2023). *NLTK*. [Online]. Available: http://www.nltk.org/api/nltk.sentiment.html

[78] (2023). *StanfordCoreNLP*. [Online]. Available: https://github.com/stanfordnlp

[79] (2023). *Alchemy*. [Online]. Available: https://docs.alchemy.com/reference/api-overview

[80] M. R. Islam and M. F. Zibran, "Leveraging automated sentiment analysis in software engineering," in *Proc. IEEE/ACM 14th Int. Conf. Mining Softw. Repositories (MSR)*, May 2017, pp. 203–214.

[81] Z. Chen, Y. Cao, X. Lu, Q. Mei, and X. Liu, "SEntiMoji: An emoji-powered learning approach for sentiment analysis in software engineering," in *Proc. 27th ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, Aug. 2019, pp. 841–852.

[82] M. R. Islam and M. F. Zibran, "A comparison of software engineering domain specific sentiment analysis tools," in *Proc. IEEE 25th Int. Conf. Softw. Anal., Evol. Reengineering (SANER)*, Mar. 2018, pp. 487–491.

[83] G. Uddin, Y.-G. Guéhénuc, F. Khomh, and C. K. Roy, "An empirical study of the effectiveness of an ensemble of stand-alone sentiment detection tools for software engineering datasets," *ACM Trans. Softw. Eng. Methodol.*, vol. 31, no. 3, pp. 1–38, Jul. 2022.
[84] T. Zhang, I. Clairine Irsan, F. Thung, and D. Lo, "Revisiting sentiment analysis for software engineering in the era of large language models," 2023, *arXiv:2310.11113*.
[85] C. Xu, Q. Sun, K. Zheng, X. Geng, P. Zhao, J. Feng, C. Tao, and D. Jiang, "WizardLM: Empowering large language models to follow complex instructions," 2023, *arXiv:2304.12244*.
[86] (2023). *Vicuna*. [Online]. Available: https://lmsys.org/blog/2023-03-30-vicuna/
[87] C. Xu, Q. Sun, K. Zheng, X. Geng, P. Zhao, J. Feng, C. Tao, and D. Jiang, "Wizardlm: Empowering large language models to follow complex instructions," 2023, *arXiv:2304.122441*.
[88] M. Mohammad Imran, P. Chatterjee, and K. Damevski, "Uncovering the causes of emotions in software developer communication using zero-shot LLMs," in *Proc. IEEE/ACM 46th Int. Conf. Softw. Eng. (ICSE)*, Apr. 2024, pp. 2244–2256.
[89] E. Biswas, M. E. Karabulut, L. Pollock, and K. Vijay-Shanker, "Achieving reliable sentiment analysis in the software engineering domain using BERT," in *Proc. IEEE Int. Conf. Softw. Maintenance Evol. (ICSME)*, Sep. 2020, pp. 162–173.
[90] Y. Chae and T. Davidson, "Large language models for text classification: From zero-shot learning to fine-tuning," *Open Sci. Found.*, vol. 1, pp. 1–15, Oct. 2023.
[91] M. M. Imran, Y. Jain, P. Chatterjee, and K. Damevski, "Data augmentation for improving emotion recognition in software engineering communication," in *Proc. 37th IEEE/ACM Int. Conf. Automated Softw. Eng.*, Oct. 2022, pp. 1–13.
[92] (2024). *Fine-Tuning GPT Models*. [Online]. Available: https://platform.openai.com/docs/guides/fine-tuning

MD SHAFIKUZZAMAN received the B.S. degree in computer science and engineering from Chittagong University of Science and Technology, Chattogram, Bangladesh, in 2015. He is currently pursuing the M.S. degree in computer science with Lamar University, Beaumont, TX, USA. His research interests include software engineering, software security, big data, cloud computing, the IoT, cyber security, and machine learning.

MD RAKIBUL ISLAM received the Ph.D. degree from The University of New Orleans, LA, USA, in 2020. He is currently an Assistant Professor with the Computer Science Department, Lamar University, Beaumont, TX, USA. His research interests include human-aspects in software engineering, software security, source code analysis, and natural language processing.

ALEX C. ROLLI is currently pursuing the B.S. degree in computer science with the University of Wisconsin–Eau Claire, WI, USA. She is also a Technical Assistant Intern with Jamf and a Research Assistant Intern with the AI and Bioinformatics Department, Mayo Clinic. Her research interests include software engineering, bioinformatics, and machine learning.

SHARMIN AKHTER received the B.S. degree in electrical and electronic engineering from the University of Information Technology and Sciences (UITS), Dhaka, Bangladesh. She is currently pursuing the M.S. degree in computer science with Lamar University, TX, USA. Her research interests include software engineering, software security, data science, cloud computing, and machine learning.

NAEEM (JIM) SELIYA received the M.S. degree in computer science and the Ph.D. degree in computer engineering from Florida Atlantic University, Boca Raton, FL, USA. He is an Assistant Professor of computer science with the University of Wisconsin–Eau Claire, WI, USA. His research interests include artificial intelligence, machine learning, deep learning, social media analytics, software engineering, cyber security, healthcare data analytics, and computing sciences education.