Pushing Feelings: Emotion and Sentiment in Software Commit Messages

Krishno Dey*, Jagannath Singh†, Hung Cao‡, Francis Palma*
* *SE+AI Research Lab, Faculty of Computer Science, University of New Brunswick, NB, Canada*
† *School of Computer Engineering, Kalinga Institute of Industrial Technology, India*
‡ *Analytics Everywhere Lab, Faculty of Computer Science, University of New Brunswick, NB, Canada*
{krishno.dey, hcao3, francis.palma}@unb.ca, jagannath.singhfcs@kiit.ac.in

2025 IEEE International Conference on Collaborative Advances in Software and COmputiNg (CASCON) | 979-8-3315-9948-5/25/$31.00 ©2025 IEEE | DOI: 10.1109/CASCON66301.2025.00072

***Abstract*—Software repositories are the primary source of information for software artifacts and contain a large amount of unstructured data, including commits, issue reports, and code comments. Mining information for different tasks, such as sentiment and emotion analysis, has been studied on several artifacts. While significant progress has been made in sentiment analysis on various artifacts, the study of emotion analysis remains under-researched due to a lack of resources, such as relevant and labeled datasets. This study presents a manually annotated dataset comprising 12,005 commit messages from the open-source Apache Tomcat project, exploring emotion and sentiment analysis through the application of natural language processing and machine learning techniques. We studied traditional models (SVM and random forest), deep learning models (bidirectional-LSTM), and pre-trained language models (LMs), as well as their ensemble, to evaluate and compare their performance on the curated dataset. Our findings suggest that software engineering domain-specific pretrained LMs consistently outperform traditional and deep learning models. The ensemble of pretrained LMs performs better on sentiment analysis. Additionally, identifying sentiment from commit messages outperforms emotion analysis tasks. We have made model implementations and the curated dataset available.**

***Index Terms*—Emotion and sentiment analysis, Pretrained language model, Ensemble learning, Commit messages.**

I. Introduction

Sentiment analysis in software engineering is crucial for identifying developer sentiments or emotions [1] expressed through various software artifacts. Although software development is a logical and systematic endeavor, the individuals behind it bring emotions, perspectives, and experience. Understanding the underlying emotions and sentiments of software artifacts is crucial for understanding the impact of sentiment and emotion on software qualities, work environment, and team dynamics. Analyzing emotions and sentiments in commit messages may provide early indications of developers' state of mind, which team leads and managers can leverage for resource allocation.

Software repositories, such as GitHub and GitLab, contain a vast amount of unstructured data, and mining this data has become inevitable for the software engineering community. Most developers' working notes on their projects are documented in commit messages, code comments, and project documentation. Although analyzing and reviewing code has received more attention from the software engineering research community [2], [3], analyzing commit messages remains challenging due to the inconsistent and informal nature of these messages [4]. Most prior studies include lexicon-based approaches [5], [6]. As such, words or lexicons played a crucial role in identifying emotions, rather than relying on contextual information. The advancement of natural language processing and machine learning techniques has paved the way for groundbreaking research in identifying emotions from software artifacts. Researchers have conducted several studies to identify emotions in commit messages using traditional machine learning approaches [7].

There has been a growing research interest in sentiment analysis on software artifacts [8]–[10]. However, due to several challenges, very few studies (e.g., [1]) focus on emotion identification. One of the main challenges is the lack of well-designed annotation guidelines, which may result in unreliable [11] and low-agreement datasets that cause biases in learning. The content of an artifact (e.g., commit messages and code comments) sometimes belongs to multiple classes [12], making it difficult to categorize them into a single class and posing challenges for the model to differentiate among classes. Another challenge is distinguishing between different emotions in similar content within the project. Developers often express their emotions and sentiments through commit messages, issue responses, code comments, and team conversations; however, it is challenging to determine sentiment and emotion from most artifacts.

Further, the performance of the models used to classify sentiment and emotion varies across the datasets [12]. Although a few open-source datasets for emotion and sentiment analysis [5], [8], [9], [13] exist in the literature, most are annotated based on the lexicons and lack manual validation. The limitations of the methods and models in capturing the underlying interpretations from texts make it difficult to analyze sentiment and emotion.

In this study, we developed an annotation guideline to facilitate emotion and sentiment analysis in commit messages. Following this guideline, we created the largest manually annotated dataset. The dataset has undergone several iterations of pre-processing and validation to make it usable for qualitative research and emotion and sentiment analysis tasks. We define two research questions to examine the state-of-the-art models

979-8-3315-9948-5/25/$31.00 ©2025 IEEE
DOI 10.1109/CASCON66301.2025.00072

for automatic emotion and sentiment analysis of software commit messages.

RQ1: How well do state-of-the-art machine learning and deep learning predict emotions and sentiments on commit messages?

RQ2: How does an ensemble of pretrained language models perform on emotion and sentiment data?

We present a comparative analysis using pretrained language models (LMs), including classical and deep learning models. The performance of the models shows that the pretrained language models outperform deep learning models, while classical models show a notable performance on this dataset. Our contribution can be summarized as follows: (1) we built the largest manually annotated dataset for emotion analysis on commit messages; (2) we investigated the pretrained LMs by fine-tuning the models with the data; (3) we investigated ensemble learning using pretrained LMs to compare and improve the performance of individual pretrained LM; (4) we are the first to provide a comparative analysis among the classical, deep learning, pretrained models, and ensemble learning; and (5) we provided a comparison of the performances between emotion and sentiment tasks.

The summary of our findings includes (1) fine-tuned software engineering domain-specific LMs provide better results compared to other models; (2) the pretrained model, CommitBART [14], trained on commit messages yields the best performance for emotion task while ensemble learning provides the best performance on sentiment task; (3) all the models except CommitBART and random forest struggled to predict the surprise class, while XLM-RoBERTa [15] only predicted neutral and trust classes for emotion analysis; and (4) all the models are performing better on sentiment tasks due to the dimensionality of the classes.

For the rest of the paper, Section II discusses an overview of related work. Section III presents our research method and the implementation strategies employed. Section IV presents a detailed performance analysis of the two experimental strategies, along with the proposed ensemble approaches, while Section V discusses the performance analysis and challenges. Finally, Section VI concludes the paper.

II. Related Work

Sentiment analysis enables one to identify the emotions from texts, which helps variously, i.e., quality of products [16], identifying requirements from user feedback [17], relating sentiment to the developer performance [18], and trust among team members [19]. Developers express their emotions and sentiments during the software development process through commit messages, issue comments, and other software artifacts [1]. Therefore, researchers became interested in extracting emotions and sentiments from software artifacts.

Researchers have been attempting to extract emotions and developing lexicon-based tools to identify both sentiment and emotions in software artifacts since the early 2010s [6], [20], [21]. As a result, researchers have developed several lexicon-based tools. Most of the lexicon tools, i.e., SentiStrength [22], and NLTK [23] have been adopted from general texts, where some tools, i.e., SentiStrength-SE [8], Senti4SD [24], and SentiSW [9] were designed for software engineering text.

A. Sentiment Analysis

The simplest sentiment analysis task involves classifying a text into one of two categories (Positive or Negative) or three categories (Positive, Negative, or Neutral). To extract developer sentiment, the text must be from sources where developers communicate or update their daily tasks. As a result, GitHub is one of the largest information repositories, containing various software artifacts, such as commit messages, code comments, and issues. Although most of the commit messages collected from GitHub belong to neutral classes [25] due to open-source collaboration, it is still worthwhile to investigate developer sentiment.

The early studies in this area rely on a lexicon-based approach to extract sentiment, which involves developing lexicons, such as SentiStrength [22] and Senti4SD [24]. Although the SentiStrength lexicon was mainly developed from social media texts, it gained popularity among researchers due to its performance in software engineering. The use of lexicons involves analyzing the relationship between programming languages and developer sentiments, as well as changes in sentiment across the days of a week and developer sentiment over a day [21].

B. Emotion Analysis

Researchers have done a few studies, e.g., [1], [10], [13] to identify the emotions[1] from commit messages and issue comments. Although emotions are categorized into eight types, researchers have used three, four, six, and eight types of emotions in the literature [1]. Lexicon-based approaches are also followed in extracting emotions, which include a few emotion lexicons, i.e., NRC Word-Emotion Association Lexicon [10], DEVA [13], TensiStrength [26], etc. Using these lexicons results in different types of emotion over a single text. For example, the NRC word-emotion association lexicon, DEVA, and TensiStrength have eight, four, and three types of emotions, respectively. Although the emotions were extracted using a lexicon, in most cases, only one word is used to identify the emotion of a text [10]. The study in [13] proposed an automatic tool named DEVA to extract emotions from commit messages and issue comments using the valence-arousal model, which calculates four types of emotions. Although the study provides an insightful tool for emotion extraction, the comparison with other tools was ambiguous.

Machine learning-based approaches have recently been studied to extract developer sentiments for software artifacts, e.g., [7]. Support vector machines, naive Bayes, random forests, decision trees, and extreme gradient boosting are among the most popular algorithms studied. A machine learning-based sentiment extraction tool was created in the study by Islam et al. [7], which outperforms existing lexicon-based sentiment extraction tools. Imran et al. [27] primarily

[1] anger, fear, anticipation, trust, surprise, sadness, joy, and disgust

addressed the data scarcity problem in emotion analysis in SE text by automatically creating new training data using a data augmentation technique by analyzing the errors made by popular SE-specific emotion classification tools. The authors investigated the effectiveness of an emotion classifier, SEnti-Moji [28], in detecting emotions in GitHub comments.

C. Emotion Analysis using LLMs

Among the very few studies that investigated large language models (LLMs) for emotions prediction in SE text, like issue comments, include [29], [30]. In [29], the authors explored zero-shot LLMs for detecting emotions in software developer communication. Those LLMs are pre-trained on massive datasets but not fine-tuned for detecting emotions in SE. The authors found that the LLMs perform well in the emotion classification task compared to state-of-the-art models. However, this comes with a significant cost in terms of computation required. Topal et al. [30] explored the performance of open-source LLMs in sentiment analysis. The authors also explored the impact of various instruction methods and fine-tuning techniques on the models' performance and analyzed their sensitivity to different prompts. The authors considered BERT-based models as their baseline and found Llama3 and Mistral to be top-performing in sentiment analysis tasks.

Existing studies mostly rely on a lexicon-based approach to extract sentiment. Moreover, there is a scarcity of datasets available for evaluating emotion and sentiment from commit messages. In this study, we construct a manually annotated dataset and employ machine learning and pre-trained language models to analyze sentiment and emotion.

III. Research Methodology and Implementation

This section details the research methodology and implementation as depicted in Figure 1.

A. Data Collection

To collect commit messages, we choose Apache Tomcat[2]. During the data collection, we collected commit IDs, committer usernames, dates, and messages. We selected the Tomcat open-source system because it is widely known, actively maintained, and used in the empirical Software Engineering research community. Tomcat has a well-documented commit history. Moreover, we also did not want to select a large or small system, opting instead for a moderate-sized dataset to be annotated. Our primary focus was to select an open-source project with numerous commit messages from developers who have contributed over a long period. We collected a total of 13,000 recent commit messages. We opted to use PyDriller[3] to extract commit messages from GitHub since it is well-documented and provides all the necessary information related to commit messages. Although each commit is unique with its commit ID, the messages can be the same, and each message can be found in multiple commits; for example, we found that `fix typo` occurs in multiple commits. We removed the duplicate commit messages to handle the duplicates and URLs from the commit messages without further preprocessing. These filtering and removing duplicate steps resulted in 12,005 commit messages.

[2]https://github.com/apache/tomcat

[3]http://dl.acm.org/citation.cfm?doid=3236024.3264598

B. Filtering Commit Messages

Commit messages contain URLs, commit IDs (i.e., hash values), and invisible characters. As a part of the preprocessing steps, we first removed the noisy portion (commit IDs, invisible characters, and blank spaces) of the data. We then removed a specific word (*git-svn-id*) due to its high frequency and lack of meaningful relevance to the emotion identification task. We also tokenized and performed case conversion (i.e., converting all characters to lowercase) before training.

C. Manual Annotation

Given a commit message, we want to categorize its intended emotion, i.e., whether it is *anger, fear, joy, neutral, sadness, surprise, or trust*.

Data Annotation Scheme: As the data annotation scheme for emotions, we adopted six emotion classes, merging disgust and anticipation with anger suggested by Shaver et al. [31] from the wheel of eight emotions [32]. For a significant number of commit messages, it is challenging to find the appropriate emotion label from the six classes. As a result, we also introduced a *neutral* class to tackle the annotation issue, resulting in seven emotion classes.

In the following, we discuss the taxonomy of emotions we provide during the annotation.

Anger is an intense emotion that intends to show irritability, disgust, envy, jealousy, and annoyance. It deliberately expresses a strong, uncomfortable, and non-supportive response to a statement.

Fear is an unpleasant emotion expressed during a dangerous or bad situation to show nervousness, anxiety, worry, panic, and dread.

Joy is a positive emotion to show happiness, satisfaction, pleasure, and amusement. The purpose of joy is to express the well-being and success of an action.

Sadness is a negative emotion with a sense of depression, regret, embarrassment, rejection, and sympathy.

Surprise is an emotion that expresses positivity to show amazement and astonishment.

Trust is an emotion of a strong belief in reliability to show confidence, admiration, and agreement with a statement.

Neutral is used for texts not representing emotions or the annotators are unable to understand the emotion from the text.

Data Annotation Guideline: We asked annotators to read the given commit message first to understand the context of the text and the emotion expressed by the writer of the message. Based on the *Data Annotation Scheme*, the annotator decides the appropriate emotion class once the annotator understands the expressed emotion. The detailed annotation guideline and the supplemental materials are available at https://doi.org/10.5281/zenodo.14449706.

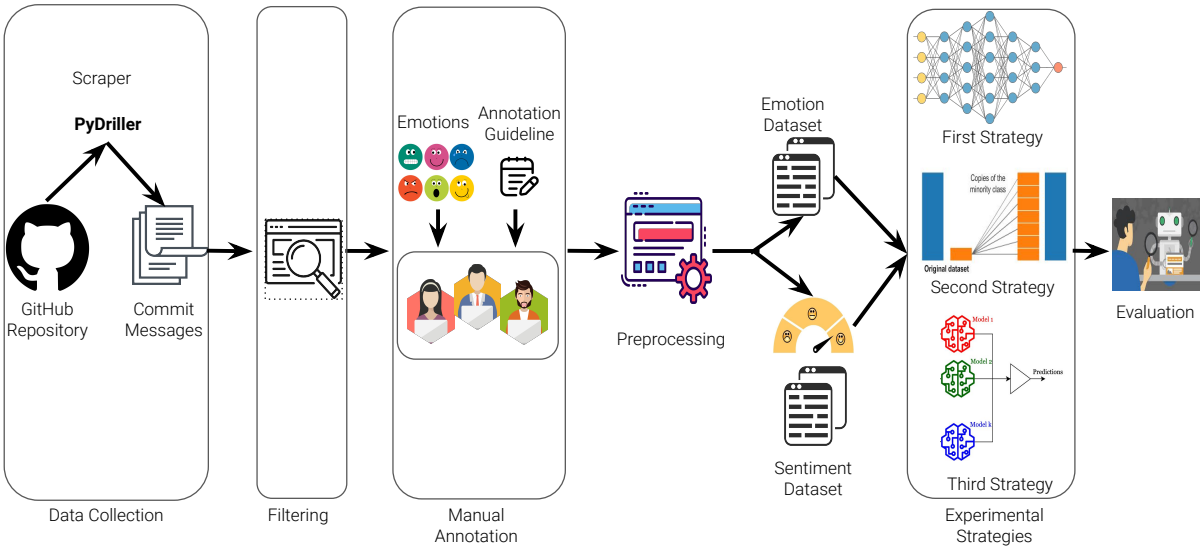

Fig. 1: The overview of the research methodology.

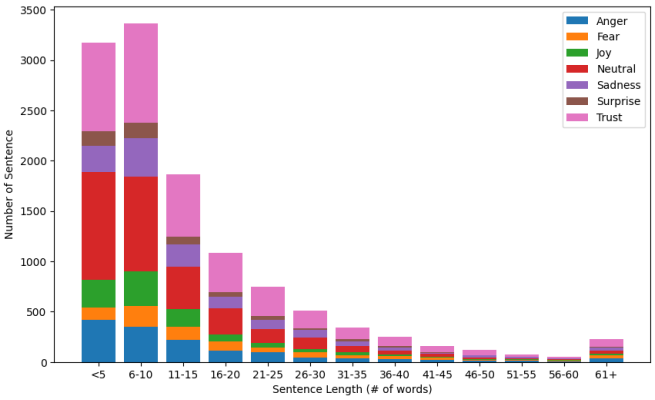

Fig. 2: The distribution of sentence length associated with each emotion label.

Annotation Process: We used a pool of annotators consisting of 14 computer science undergraduate and graduate students. All of the annotators are divided into two groups: annotators and curators. In the annotators group, we selected 12 students; the rest (those with previous data annotation experience) were used as curators. Students in the annotator groups had low to moderate experience in data annotation and sentiment/emotion analysis from text. In contrast, researchers in the curators group were experienced in data annotation and sentiment/emotion analysis in the software engineering domain. The primary task of the annotator group was to annotate the commit messages, while the curators concentrated on consolidating the final labels. Each commit message was independently annotated by three annotators, following our annotation guidelines that describe the annotation steps. We conducted a weekly review meeting to align and discuss specific annotation cases. The majority agreement for each commit message selected the final label. However, in cases of disagreement among the annotators, the curators organized a discussion to select the final label. This method offers high accuracy, which is critical for any annotation task. However, this process is costly and time-consuming for larger datasets.

Annotation Quality: We calculated the Inter-Annotator Agreement (IAA) to measure the annotation quality. First, we calculated the Fleiss Kappa (κ) [33] score on the unconsolidated labeled dataset, yielding a value of 0.14, indicating *slight agreement* among the annotators. This annotation score indicates the disagreement among annotators for a large number of data points, i.e., there was no majority label for emotions among the annotators. The curators consolidated the final labels for the commit messages with disagreement. Afterward, we again calculated the κ score and obtained a value of 0.52, indicating a *moderate agreement* among annotators with the help of the curator's group.

D. Pre-processing

We pre-process the data by removing unnecessary text and prepare it for training and evaluation.

1) Data Analysis: We analyzed the sentence distributions by the number of words for each class label presented in Figure 2. Based on the distribution, *neutral* and *trust* are the most common emotions developers express in commit messages, i.e., around 58% of the data falls within the trust and neutral classes. *Surprise* and *fear* are the least occurring emotions in our dataset, comprising around 11% of the dataset. We also incorporate different ranges of sentence length buckets to investigate the optimal sequence length for pretrained LMs. We found that around 80% data belongs within the sentence length of 20 words.

2) Data Split: We divided the dataset into train, validation, and test splits, containing 70%, 10%, and 20% of the data, respectively. We used stratified sampling to ensure a balanced

Class	Train	Dev	Test	Total	Class	Train	Dev	Test	Total
Anger	953	156	276	1,385	Sadness	929	124	273	1,326
Fear	538	91	173	802	Surprise	365	51	111	527
Joy	740	89	204	1,033	Trust	2,678	367	739	3,784
Neutral	2,201	310	637	3,148					

TABLE I: Class label distribution across the data splits.

distribution of class labels across the data splits. We provided the detailed data distribution of the data splits in Table I.

3) Sentiment Data Preparation: *Anger*, *Fear*, and *Sadness* emotions are expressed in a negative situation while *Joy* and *Trust* show positivity. Moreover, *Surprise* emotion can be expressed in positive and negative situations. However, we chose *Surprise* emotion as a positive sentiment since we found positive textual representations in the commit messages. As a result, to consolidate the sentiment data, we combined {*Anger*, *Fear*, *Sadness*} emotions as *Negative* sentiments and {*Joy*, *Surprise*, *Trust*} emotions as *Positive* sentiments while considering neutral emotions as neutral sentiments.

E. Experimental Strategies

We employed two experimental strategies to investigate the two research questions.

1) First Strategy: As part of this strategy, we address RQ1 to compare the performance of traditional machine learning, deep learning, and pretrained LMs using both emotion and sentiment datasets. Indeed, we selected widely used models effective for sentiment and emotion analysis.

Baseline Model: We incorporate multinomial naive Bayes (MNB) as our baseline, as it has been widely considered the standard model in the literature. We capture contextual information for MNB by using a weighted $n-$gram (uni, bi, tri, and quadri-gram).

Classical Models: In prior studies, classical machine learning models such as the support vector machine (SVM) and random forest have been widely used due to limited computational resources [13].To investigate classical machine learning models, we selected SVM and RF to conduct our experiments. We used the standard parameter settings and a weighted $n-$gram tf-idf representation to prepare the data.

Deep Learning: Although deep learning models are extensively used for sentiment and emotion analysis on social media data, they have not been widely applied to sentiment or emotion identification for software artifacts to date. We incorporate bidirectional long short-term memory (LSTM) as a deep learning algorithm to explore the performance of the deep learning model. The motivation for choosing LSTM is that it can capture a long sequence and extract meaningful information. We use an embedding dimension of 128, as more than 95

Pretrained Language Models: We choose `XLM-RoBERTa` [15], `CodeBERT` [2], `CodeReviewer` [3], T5 commit generator[4], and `CommitBART` [14] pretrained models for our study because transformer-based pretrained LMs have exhibited notable performance for various NLP downstream tasks. We maintain the same hyper-parameters for fine-tuning for fair evaluation across the models. We fine-tune our models for up to 3 epochs to mitigate overfitting, using an Adam optimizer with a learning rate of $2e^{-5}$, a batch size of 16, and a maximum sequence length of 256.

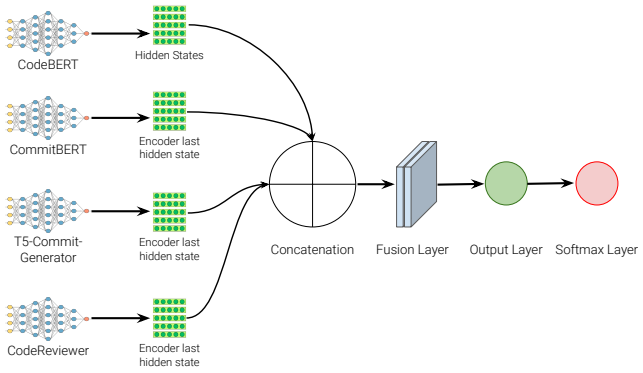

Fig. 3: Concatenation of hidden outputs followed by fusion and output layer.

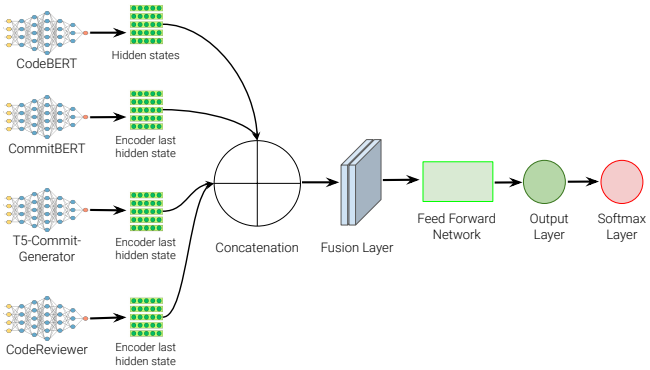

Fig. 4: Concatenation of hidden outputs followed by fusion layer, feed-forward network, and output layer.

2) Second Strategy: To address RQ2, we explore three different approaches for ensemble learning. We utilize four different pre-trained LMs, trained on software artifacts, in all three approaches. In the first approach, we concatenate the output of the hidden state of CodeBERT and the output of the encoder's last hidden layer of the other three models (CommitBART, T5 Commit Generator, and CodeReviewer). Then, we feed the concatenated output to the fusion layer, followed by the output and softmax layer. For the second approach, we maintain a similar architecture to the first approach and incorporate a feed-forward network between the fusion and output layers. The primary difference between the first and second approaches lies in the softmax layer. For the third approach, we select the best logits based on the logits confidence and feed the best logits to the softmax layer. We

[4]https://huggingface.co/mamiksik/T5-commit-message-generation

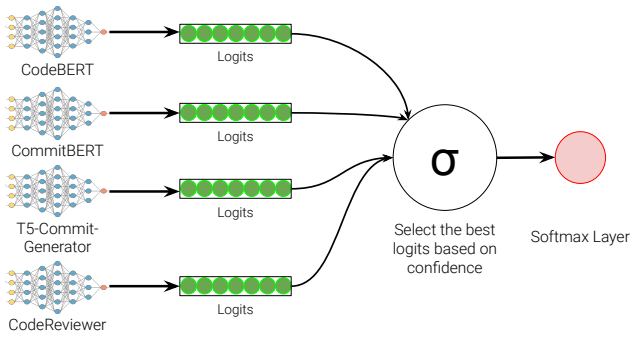

Fig. 5: Third ensemble approach: select the best logits based on the logits' confidence.

provide a detailed architecture of the first, second, and third approaches in Figures 3, 4, and 5, respectively. We train all the ensemble models for up to 10 epochs with a learning rate of $2e^{-5}$ for the Adam optimizer, a batch size of 32, and a maximum sequence length of 256. Moreover, we used the `CrossEntropyLoss` function to calculate the training and validation loss.

To evaluate and compare the models, we computed their accuracy, weighted precision, recall, and F1 macro score to measure all performance across the different experimental settings. Considering the imbalanced data, we chose the macro version of the F1 score.

IV. Experimental Results

This section details the results of the two strategies addressing their corresponding research questions.

A. RQ1: Performance of Machine and Deep Learning Models

In Table II (top), we present the results of our first experiment strategy with the original train set of the emotion dataset. In our first experiments, we found that all the models outperformed the baseline, except for the XLM-RoBERTa-large model. Classical models outperform the baseline and XLM-RoBERTa-large model, while the random forest outperforms the bidirectional LSTM and T5 commit generator. While most pre-trained LMs outperform classical and deep learning models, XLM-RoBERTa-large failed to demonstrate superior performance on commit messages. The reason for performing poorly on commit messages is that the model is trained on newspapers, Wikipedia, and formal text, which differs from commit messages. Among the models trained on software artifacts, CommitBART is the best-performing model, while CodeBERT shows prominent performance over the test set.

We present the results on the sentiment dataset for the first strategy in Table II (bottom). This strategy used original training data to train the model and test data for evaluation. In our first experimental strategy, we found that all the models outperformed the baseline except the XLM-RoBERTa-large model. Classical models provided comparable performances, while the random forest outperformed bidirectional LSTM, CodeReviewer, and the T5 commit generator. Moreover, the bidirectional LSTM outperforms two pre-trained

Model	Accuracy	Precision	Recall	F1-macro
Baseline				
MNB	39.66	35.74	39.66	21.43
Classical Models				
SVM	38.25	35.16	38.25	**26.51**
RF	42.27	40.00	42.27	**28.65**
Deep Learning Model				
Bidirectional LSTM	38.50	35.75	38.50	**27.42**
Pretrained Language Models				
XLM-RoBERTa-large	37.05	21.34	37.05	13.56
CodeBERT	43.72	44.06	43.72	**30.71**
CodeReviewer	42.40	41.48	42.40	**29.97**
CommitBART	43.72	42.96	43.72	**<u>32.70</u>**
T5 Commit Generator	42.77	39.16	42.77	**28.11**

Model	Accuracy	Precision	Recall	F1-macro
Baseline				
MNB	51.45	51.32	51.45	46.51
Classical Models				
SVM	51.20	50.65	51.20	**49.06**
RF	54.61	54.49	54.61	**51.57**
Deep Learning Model				
Bidirectional LSTM	51.99	51.61	51.99	**50.07**
Pretrained Language Models				
XLM-RoBERTa-large	43.73	19.12	43.73	20.28
CodeBERT	55.52	55.19	55.52	**<u>53.36</u>**
CodeReviewer	52.99	53.36	52.99	**49.29**
CommitBART	54.03	53.84	54.03	**51.77**
T5 Commit Generator	52.49	52.14	52.49	**49.89**

TABLE II: Performance on different models for the first experiment strategy to address RQ1 using emotion (top) and sentiment (bottom) datasets. **Bold** indicates models outperformed baseline; <u>Underline</u> indicates the best performing model.

LMs (CodeReviewer and T5-Commit-Generator) despite having less model complexity and fewer trainable parameters. However, CodeBERT and CommitBART exhibit superior performance compared to the other models, with CodeBERT being the best-performing model, achieving an F1-macro score of 53.36.

Therefore, the current state-of-the-art machine learning, deep learning, and pre-trained LMs did not perform well for emotion analysis, most likely due to the high number of emotion classes (i.e., seven) and poor annotation quality. However, the performance on the sentiment dataset (with only three classes) is comparable to state-of-the-art results.

B. RQ2: Performance of Ensemble Learning

Table III (top) presents the performance of the second strategy using the emotion dataset. The performance of ensemble learning did not perform well on the emotion dataset. We achieved an F1-macro score of 20.26% for the first ensemble approach, which could not surpass the baseline. Although the other two ensemble approaches outperformed the baseline, the pretrained LMs (trained on software artifacts) and random

Model	Accuracy	Precision	Recall	F1-macro
Baseline	39.66	35.74	39.66	21.43
Approach 1	41.45	28.82	41.45	20.26
Approach 2	37.75	36.29	37.75	**27.62**
Approach 3	39.45	36.88	39.45	**28.30**
Model	**Accuracy**	**Precision**	**Recall**	**F1-macro**
Baseline	51.45	51.32	51.45	46.51
Approach 1	56.15	56.16	56.15	**54.67**
Approach 2	54.82	54.52	54.82	**52.85**
Approach 3	54.98	55.05	54.98	**52.35**

TABLE III: Performance on proposed ensemble techniques to address RQ2 for emotion (top) and sentiment (bottom) datasets.

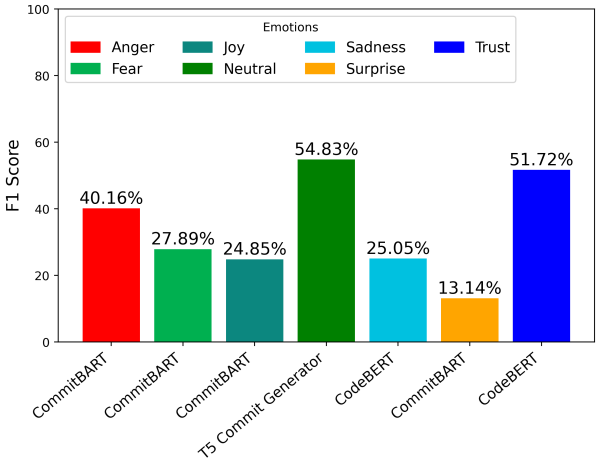

Fig. 6: Emotion class-wise best-performing models.

forest exhibit superior performances in emotion analysis. We achieved F1-macro scores of 27.62% and 28.30% for the second and third ensemble approaches, respectively.

Table III (bottom) presents the performance of the third strategy on the sentiment dataset. Despite the poor performance on the emotion dataset, the third strategy delivers good performance on the sentiment dataset. The first ensemble approach outperformed all the experiments with an F1-macro score of 54.67%, while the other two ensemble approaches outperformed all the models except CodeBERT. We achieved F1-macro scores of 52.85% and 52.35% for the second and third ensemble approaches. Therefore, the performance on the emotion dataset after applying ensemble learning decreased by $4\% - 12\%$. However, ensemble learning provides the best performance on the sentiment dataset. The observed decrease in performance for the emotion dataset and the slight improvement for the sentiment dataset suggest that ensemble learning is not always beneficial, and its effectiveness may vary depending on the chosen approach.

V. Discussions

For the first strategy, all the models struggled to predict the *Fear*, *Joy*, and *Sadness* classes, while the prediction for the

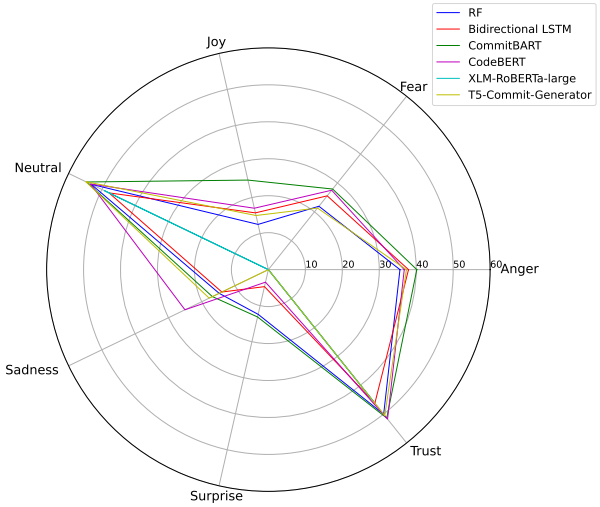

Fig. 7: Class-wise performances on selected models for the first experiment strategy on the emotion dataset.

Surprise class was the most challenging for emotion analysis. The XLM-RoBERTa-large model correctly predicted only the *Neutral* and *Trust* classes, but failed to predict the other classes correctly. Bidirectional LSTM performed poorly in the *Surprise* class while struggling with the *Sadness* and *Joy* classes. The pretrained model T5 Commit Generator could not predict any *Surprise* class correctly while delivering the best performance for the *Neutral* class and the second-best performance for the *Trust* class. While CodeBERT performed inadequately in the *Surprise* class, it achieved the best performance for the *Sadness* and *Trust* classes and the second-best performance in the *Fear* class. Among all the models, CommitBART outperformed the others in four classes: *Anger*, *Fear*, *Joy*, and *Surprise*. We provided the detailed class-wise performances on selected models in Figure 7. We present class-wise best-performing models for emotion analysis in Figure 6.

Figure 8 and Table IV present the detailed class-wise performance results for the selected models. We also analyzed the class-wise performance for the first experimental strategy on the sentiment dataset. We noticed that all the models performed better, except for the XLM-RoBERTa-large model, compared to the emotion dataset performance. The XLM-RoBERTa-large model predicts only the positive class. However, the classical model and random forest show the best performance on the *Negative* and *Neutral* classes, whereas CodeBERT exhibits superior performance on the positive class. Moreover, the performance on the sentiment dataset is superior to that on the emotion dataset; specifically, the higher number of classes in the emotion dataset leads to poorer performance. We expect that a well-balanced and better human-annotated dataset will yield good performance for both emotion and sentiment analysis, as evidenced by the results.

We explored three different ensemble learning approaches to understand the performance of emotion and sentiment analysis.

TABLE IV: Class-wise performances on selected models for the first experiment strategy on the emotion dataset. **Bold** indicates the best performance for the class.

Model Name	Class	F1 score	Model Name	Class	F1 score
Random Forest	Anger	35.60	XLM-RoBERTa-large	Anger	0.00
	Fear	21.93		Fear	0.00
	Joy	12.55		Joy	0.00
	Neutral	53.24		Neutral	49.19
	Sadness	14.78		Sadness	0.00
	Surprise	12.41		Surprise	0.00
	Trust	50.03		Trust	45.71
Bidirectional LSTM	Anger	37.92	T5 Commit Generator	Anger	37.34
	Fear	25.53		Fear	21.31
	Joy	15.75		Joy	15.02
	Neutral	47.85		Neutral	**54.83**
	Sadness	13.97		Sadness	17.78
	Surprise	4.72		Surprise	0.00
	Trust	46.21		Trust	50.51
CommitBART	Anger	**40.16**	CodeBERT	Anger	36.76
	Fear	**27.89**		Fear	27.48
	Joy	**24.85**		Joy	17.04
	Neutral	54.65		Neutral	53.41
	Sadness	16.82		Sadness	**25.05**
	Surprise	**13.14**		Surprise	3.48
	Trust	51.36		Trust	**51.72**

TABLE V: Example of contradictory data annotation in the training set.

hash	Commit Message	Class
d4c330cfdc2610f86d1cbfda8244797461804577	Fix typos	Trust
a15025606e31db384d1ec23f5ac17c4a49bd9579	Fix typo	Neutral
cea7a21e61fe55220bda12ace665c33028ed2d02	Fix name typo	Sadness
2704d89e3aa279bf791ef4a7c793a30d5d53e55c	Correct a typo	Joy

In our experiments, the ensemble learning approaches did not perform as expected on the emotion dataset, whereas they exhibited superior performance on the sentiment dataset. This demonstrates that ensemble learning is effective for performance improvement in sentiment analysis. For the emotion dataset, the first ensemble approach only predicts the *Anger*, *Neutral*, and *Trust* classes, while it provides the best performance on the sentiment dataset. The performance on the emotion dataset significantly decreased due to the large number of emotion classes compared to the sentiment dataset.

Although deep learning and pre-trained LMs provide comparative results, their performances are below 33% for emotion analysis in our study. We investigated the data quality to get a more insightful conclusion. We found that the manual annotation process yielded conflicting labels for similar kinds of data. Table V presents an example of contradictory annotation from our training data. Also, due to the biases among the annotators, the quality of data annotation is flawed. Furthermore, we found that 15% of the training data consists of two or three words. Although capturing the emotions and sentiment from a sentence of two or three words is challenging, we did not remove those texts. We did not remove the two- or three-word texts to understand the performances across the models because a significant number of commit messages are of short text length.

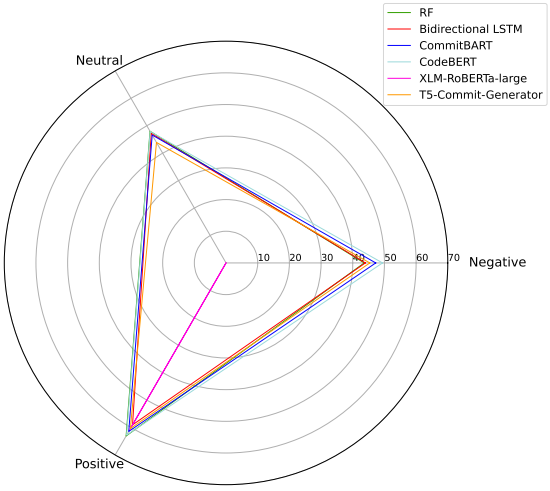

Fig. 8: Class-wise performances on selected models for the first experiment strategy on the sentiment dataset.

We identified that capturing human feelings is challenging if the text contains insufficient information. For example, fixing an important bug or feature might lead to joy (positive), whereas fixing some minor issues might lead to anger (negative). However, the commit messages for both cases are similar. We also noticed that most texts contain noise, such as class names, method names, versions, and package information. These noises affect the overall performance of the models because they occur in multiple emotion categories. Removing such noises requires manual efforts because of the unstructured representation of texts.

A. Comparison

Table VI shows a performance comparison among various studies in the literature. Some studies have not evaluated au-

Studies	Analysis	Dataset	Classes	Best F1 Score
(2014) Guzman et al. [21]	Sentiment	GitHub commit comments	2	-
(2016) Sinha et al. [25]	Sentiment	GitHub commit comments	3	-
(2017) Thelwall [26]	Emotion	Tweets	2	54.45%
(2018) Islam and Zibran [12]	Emotion	Jira issue comments	4	78.43%
(2019) Islam et al. [7]	Emotion	Jira and stack overflow comments	5	80.90%
(2021) Venigalla and Chimalakonda [10]	Emotion	Github commit comments	8	-
(2022) Imran et al. [27]	Emotion	GitHub comments	6	48%
(2024) Imran et al. [29]	Emotion	GitHub, Jira, and Stack overflow comments	6	59.20%
(2024) Topal et al. [30]	Emotion	Tweets	4	36.5%-56.3%
Our Study	Emotion	GitHub commit comments	6	32.70%
Our Study	Sentiment	GitHub commit comments	3	54.67%

TABLE VI: Comparison of Existing Studies on Emotion and Sentiment Analysis in SE.

tomated methods for identifying emotion and sentiment, e.g., [10], [21], [25]. Overall, studies considering fewer emotion or sentiment classes performed well, e.g., [12]. Depending on the dataset, the performance of machine learning and deep learning, as well as large language models, also varies significantly. Thus, benchmarking is extremely challenging in emotion and sentiment analysis in the SE domain. Early studies (e.g., [7], [12]) used traditional machine learning models, whereas more recent studies (e.g., [29], [30]) explored the ability of computationally expensive large language models and fine-tuned them. However, researchers should consider the trade-off between cost and benefits while analyzing emotion and sentiment in the software engineering domain.

B. Implications for Researchers and Developers

The literature on emotion and sentiment analysis of commit messages lacks manually annotated data. As a result, studies on emotion and sentiment analysis using commit messages have not attracted many researchers. Our study provided the largest human-annotated dataset for emotion and sentiment analysis, which may attract the research community to explore this area. Moreover, our findings suggest that a good annotation guideline is required for constructing a better dataset using commit messages. This study aims to explore developers' emotions and sentiments expressed in commit messages; thus, devising high-performing machine learning models was not the primary goal of this study.

Our study shows that developers mostly use *neutral* and *trust* emotions while writing commit messages. However, a significant number of commit messages contain *anger* and *sadness*, indicating that developers often write code with negative emotions, which can impact the project's quality. Moreover, we found that developers do not write long text ($\sim$80% of commit messages are within 20 words) in commit messages. Thus, we strongly recommend that developers write more expressive and meaningful commit messages, which offer several benefits (e.g., issue tracking), during both software development and maintenance and evolution phases. We also found that developers express different emotions without significant changes in the commit messages. The manual annotation of the dataset is highly imbalanced, which may significantly impact the model's performance. Moreover, the annotations are subjective, and the annotators base their annotations on their understanding, which may introduce biases into our dataset. Therefore, it is crucial to account for annotator bias when developing models utilizing curated datasets. The hyper-parameters we used to train our models require further investigation and experiments for optimal performance. Investigating optimal hyper-parameters was not within the scope of this study, i.e., we aimed to explore emotions and sentiments in the commit messages. Our experiments may yield better results with optimal hyper-parameters. We have made model implementations and the curated dataset available for replication.

VI. Conclusion and Future Work

This study presents the largest manually annotated dataset for emotion and sentiment analysis on commit messages and reports the performance of pre-trained language models (LMs), deep learning, traditional machine learning, and the ensemble of pre-trained LMs in predicting emotion and sentiment. Our first experimental strategy to address RQ1 showed that state-of-the-art machine learning models perform better on the sentiment dataset than the emotion dataset. The RQ2 demonstrated that the oversampling technique in our study did not perform well. Furthermore, findings in RQ2 suggested that incorporating ensemble learning improves the performance of the sentiment dataset while negatively impacting the performance of the emotion dataset. We also provided a detailed performance comparison among the models.

We conducted all the experiments using the proposed dataset and provided an in-depth analysis. Our results indicate that while the pre-trained LMs outperformed the classical and deep learning algorithms, their performance was insufficient to identify emotions accurately. Although the annotation of the dataset showed moderate agreement, the advancement of pre-trained LMs for commit messages has yet to be studied. Moreover, a detailed annotation guideline is required to construct a new dataset for this area. We also identified the challenges of determining the underlying meaning of the text, which caused low performance across the models. In addition, further study of the pre-trained model represents another promising direction, and we will incorporate new pre-trained models for commit messages in our future studies. We also aim to

develop a more diverse dataset that would better demonstrate the robustness of the proposed approach.

ACKNOWLEDGMENT

This study is supported by the New Brunswick Innovation Foundation (NBIF) through the Grant TRF2023-003. The authors would like to thank Md. Arid Hasan for initiating this work and the annotators for labeling the dataset.

REFERENCES

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

[2] Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, et al. Codebert: A pre-trained model for programming and natural languages. *arXiv preprint arXiv:2002.08155*, 2020.

[3] Zhiyu Li, Shuai Lu, Daya Guo, Nan Duan, Shailesh Jannu, Grant Jenks, Deep Majumder, Jared Green, Alexey Svyatkovskiy, Shengyu Fu, et al. Automating code review activities by large-scale pre-training. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 1035–1047, 2022.

[4] Yue Yang, Elisabetta Ronchieri, and Marco Canaparo. Natural language processing application on commit messages: a case study on hep software. *Applied Sciences*, 12(21):10773, 2022.

[5] Fabio Calefato, Filippo Lanubile, and Nicole Novielli. Emotxt: a toolkit for emotion recognition from text. In *2017 seventh international conference on Affective Computing and Intelligent Interaction Workshops and Demos (ACIIW)*, pages 79–80. IEEE, 2017.

[6] Alessandro Murgia, Parastou Tourani, Bram Adams, and Marco Ortu. Do developers feel emotions? an exploratory analysis of emotions in software artifacts. In *Proceedings of the 11th working conference on mining software repositories*, pages 262–271, 2014.

[7] Md Rakibul Islam, Md Kauser Ahmmed, and Minhaz F Zibran. MarValous: Machine Learning based Detection of Emotions in the Valence-arousal Space in Software Engineering Text. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, pages 1786–1793, 2019.

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

[9] Jin Ding, Hailong Sun, Xu Wang, and Xudong Liu. Entity-level sentiment analysis of issue comments. In *Proceedings of the 3rd International Workshop on Emotion Awareness in Software Engineering*, pages 7–13, 2018.

[10] Akhila Sri Manasa Venigalla and Sridhar Chimalakonda. Understanding emotions of developer community towards software documentation. In *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*, pages 87–91. IEEE, 2021.

[11] Nasif Imtiaz, Justin Middleton, Peter Girouard, and Emerson Murphy-Hill. Sentiment and politeness analysis tools on developer discussions are unreliable, but so are people. In *Proceedings of the 3rd International Workshop on Emotion Awareness in Software Engineering*, pages 55–61, 2018.

[12] Md Rakibul Islam and Minhaz F Zibran. A comparison of software engineering domain specific sentiment analysis tools. In *2018 IEEE 25th international conference on software analysis, evolution and reengineering (SANER)*, pages 487–491. IEEE, 2018.

[13] Md Rakibul Islam and Minhaz F Zibran. DEVA: Sensing Emotions in the Valence Arousal Space in Software Engineering Text. In *Proceedings of the 33rd annual ACM symposium on applied computing*, pages 1536–1543, 2018.

[14] Shangqing Liu, Yanzhou Li, and Yang Liu. Commitbart: A large pre-trained model for github commits. *arXiv preprint arXiv:2208.08100*, 2022.

[15] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*, 2019.

[16] Mohammad Masudur Rahman, Chanchal K Roy, and Iman Keivanloo. Recommending insightful comments for source code using crowdsourced knowledge. In *2015 IEEE 15th International Working Conference on Source Code Analysis and Manipulation (SCAM)*, pages 81–90. IEEE, 2015.

[17] Nazakat Ali, Sangwon Hwang, and Jang-Eui Hong. Your opinions let us know: Mining social network sites to evolve software product lines. *KSII Transactions on Internet & Information Systems*, 13(8), 2019.

[18] Ian Brooks and Kathleen Swigger. Using sentiment analysis to measure the effects of leaders in global software development. In *2012 International Conference on Collaboration Technologies and Systems (CTS)*, pages 517–524. IEEE, 2012.

[19] Guilherme A. Maldonado da Cruz., Elisa Hatsue Moriya Huzita., and Valéria D. Feltrim. Estimating trust in virtual teams - a framework based on sentiment analysis. In *Proceedings of the 18th International Conference on Enterprise Information Systems - Volume 1: ICEIS*, pages 464–471. INSTICC, SciTePress, 2016.

[20] Emitza Guzman and Bernd Bruegge. Towards emotional awareness in software development teams. In *Proceedings of the 2013 9th joint meeting on foundations of software engineering*, pages 671–674, 2013.

[21] Emitza Guzman, David Azócar, and Yang Li. Sentiment analysis of commit comments in github: an empirical study. In *Proceedings of the 11th working conference on mining software repositories*, pages 352–355, 2014.

[22] Mike Thelwall, Kevan Buckley, Georgios Paltoglou, Di Cai, and Arvid Kappas. Sentiment strength detection in short informal text. *Journal of the American society for information science and technology*, 61(12):2544–2558, 2010.

[23] Clayton Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the international AAAI conference on web and social media*, volume 8, pages 216–225, 2014.

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

[25] Vinayak Sinha, Alina Lazar, and Bonita Sharif. Analyzing developer sentiment in commit logs. In *Proceedings of the 13th international conference on mining software repositories*, pages 520–523, 2016.

[26] Mike Thelwall. Tensistrength: Stress and relaxation magnitude detection for social media texts. *Information Processing & Management*, 53(1):106–121, 2017.

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

[28] Zhenpeng Chen, Yanbin Cao, Xuan Lu, Qiaozhu Mei, and Xuanzhe Liu. SEntiMoji: An Emoji-powered Learning Approach for Sentiment Analysis in Software Engineering. In *Proceedings of the 2019 27th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering*, pages 841–852, 2019.

[29] Mia Mohammad Imran, Preetha Chatterjee, and Kostadin Damevski. Uncovering the Causes of Emotions in Software Developer Communication using Zero-shot LLMs. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, pages 1–13, 2024.

[30] Mustafa Burak Topal, Aysun Bozanta, and Ayse Basar. Sentiment Analysis with LLMs: Evaluating QLoRA Fine-tuning, Instruction Strategies, and Prompt Sensitivity. In *Proceedings of the 34th International Conference on Collaborative Advances in Software and COmputiNg (CASCON)*, 2024.

[31] Phillip Shaver, Judith Schwartz, Donald Kirson, and Cary O'connor. Emotion knowledge: Further exploration of a prototype approach. *Journal of personality and social psychology*, 52(6):1061, 1987.

[32] Robert Plutchik. A general psychoevolutionary theory of emotion. In *Theories of emotion*, pages 3–33. Elsevier, 1980.

[33] Joseph L Fleiss and Jacob Cohen. The equivalence of weighted kappa and the intraclass correlation coefficient as measures of reliability. *Educational and psychological measurement*, 33(3):613–619, 1973.