Uncovering the Causes of Emotions in Software Developer Communication Using Zero-shot LLMs

Mia Mohammad Imran
Virginia Commonwealth University
Richmond, Virginia, USA
imranm3@vcu.edu

Preetha Chatterjee
Drexel University
Philadelphia, Pennsylvania, USA
pc697@drexel.edu

Kostadin Damevski
Virginia Commonwealth University
Richmond, Virginia, USA
kdamevski@vcu.edu

ABSTRACT

Understanding and identifying the causes behind developers' emotions (e.g., *Frustration* caused by 'delays in merging pull requests') can be crucial towards finding solutions to problems and fostering collaboration in open-source communities. Effectively identifying such information in the high volume of communications across the different project channels, such as chats, emails, and issue comments, requires automated recognition of emotions and their causes. To enable this automation, large-scale software engineering-specific datasets that can be used to train accurate machine learning models are required. However, such datasets are expensive to create with the variety and informal nature of software projects' communication channels.

In this paper, we explore zero-shot LLMs that are pre-trained on massive datasets but without being fine-tuned specifically for the task of detecting emotion causes in software engineering: ChatGPT, GPT-4, and flan-alpaca. Our evaluation indicates that these recently available models can identify emotion categories when given detailed emotions, although they perform worse than the top-rated models. For emotion cause identification, our results indicate that zero-shot LLMs are effective at recognizing the correct emotion cause with a BLEU-2 score of 0.598. To highlight the potential use of these techniques, we conduct a case study of the causes of *Frustration* in the last year of development of a popular open-source project, revealing several interesting insights.

ACM Reference Format:
Mia Mohammad Imran, Preetha Chatterjee, and Kostadin Damevski. 2024. Uncovering the Causes of Emotions in Software Developer Communication Using Zero-shot LLMs. In *2024 IEEE/ACM 46th International Conference on Software Engineering (ICSE '24), April 14–20, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3597503.3639223

1 INTRODUCTION

Emotions play a crucial role in software engineering, influencing individual and team performance, communication, and decision-making. Numerous software engineering tasks have been found to be impacted by developer emotions, e.g., bug fixing efficiency [1, 2], build success of continuous integration [3]. Previous research has also studied the link between developers' emotions, their productivity, and their attrition in software development projects [4–6]. Open source projects' success depends on attracting and retaining volunteer participants. Automatically detecting the emotions in the communication channels of a software project, e.g., pull requests and issues comments, chats, and discussion boards, can provide valuable insights into improving the outcomes of open-source software projects.

Going beyond detecting the occurrence of different emotions in developer communication channels, identifying the causes of those emotions is key for many uses. Simply knowing the existence of an emotion is often insufficient for understanding what or whom the expressed emotion is towards and for determining the appropriate reaction [7]. However, when the emotion cause is known, along with the type of emotion, it becomes possible to reliably assess potential implications. This could allow for understanding developer opinion towards different aspects of the project like technical debt [8] or code reviews [9]. For instance, a developer made the following comment on an open issue in the flutter/flutter GitHub project, *"this is a really severe issue, the ux is pretty awful when you have a splash and then a landing page to simulate splash because it is very obvious that is a different view than the splash"*, which expresses *Frustration* (a sub-emotion of *Anger*). The cause of this emotion is the text span, *"the ux is pretty awful"*. Extracting emotion causes automatically is challenging because of the distinct nature of software engineering communication (e.g., it includes domain-specific idioms like *'spaghetti code'*), the variety of different channels (e.g., chats vs. issue comments), and the informal nature of developer communication (e.g., often containing informal abbreviations like 'AFAIK' [10]). It is likely that emotion-cause extraction requires a large amount of software engineering-specific training data that can capture this variability, in both emotion and language [11, 12].

Large Language Models (LLMs) have recently emerged as a new powerful type of deep learning technique. These models are built by unsupervised pre-training on a very large dataset, followed by supervised fine-tuning on a smaller dataset. During pre-training, the model learns to predict the next word, given a sequence of words. During fine-tuning, the model is provided with labeled data relevant to a specific task. Some of the largest and most powerful LLMs, such as ChatGPT [13] and GPT-4 [14], are now widely available but do not disclose details about their dataset, training process, or model weights. Consequently, fine-tuning them for a specific task or dataset, such as detecting emotion-causes in software engineering text, is not possible. However, these LLMs can still be used as "zero-shot" models, where no task-specific fine-tuning is performed. Since constructing a large training dataset for emotion-cause extraction task in software engineering communication is expensive, using a zero-shot setup is an attractive option. Models used in a "zero-shot" setup are sometimes referred to as "zero-shot LLMs" in the

ICSE '24, April 14–20, 2024, Lisbon, Portugal

ACM ISBN 979-8-4007-0217-4/24/04.
https://doi.org/10.1145/3597503.3639223

literature [15–18]. In this paper, we use the same phrase to refer to this model setup.

This paper applies 3 models in zero-shot setup, ChatGPT [13], GPT-4 [14], and one that is open-source, flan-alpaca [19], to the problem of emotion-cause extraction in software engineering. We first examine the ability of such models to detect emotions in software engineering text, relative to state-of-the-art techniques and to LLMs fine-tuned for detecting emotions. Next, we examine the effectiveness of the zero-shot LLMs for the emotion-cause extraction task. The results indicate that these models are promising, achieving a BLUE-2 score of 0.598 on a manually curated dataset of 450 utterances. Finally, we perform a case study on the causes of *Frustration*, an undesirable emotion within a large open-source software project [20], to further highlight the utility of emotion-cause extraction for software engineering. The main contributions of this paper are:

- application and evaluation of zero-shot LLMs to the problem of emotion-cause extraction in software engineering.
- manually-curated emotion-cause extraction dataset of 450 GitHub comments.
- case study highlighting the usefulness and purpose of automatic emotion-cause extraction in software engineering.
- evaluation of zero-shot LLMs compared to state-of-the-art techniques, including fine-tuned LLMs, on the well-known problem of classifying emotions in software engineering communication.

We publish the source code and annotated dataset to facilitate the replication of our study at: https://github.com/vcu-swim-lab/SE-Emotion-Cause-Replication.

2 PRELIMINARY STUDY: DETECTING EMOTION TYPES

Detecting the causes of emotions in text requires a reliable model that can accurately identify the type of emotion expressed. Therefore, before proceeding, we conduct a preliminary investigation to determine if zero-shot LLMs can accurately detect emotions in software engineering texts. We compare the performance of these models with 1) existing state-of-the-art emotion classification models in software engineering, and 2) fine-tuned LLMs. The models are evaluated on three different types of datasets: a) GitHub comments dataset [21], b) Stack Overflow comments dataset [22], and c) JIRA comments dataset [2].

2.1 Datasets

GitHub Dataset. Imran et al. curated a diverse collection of 2000 data points sourced from GitHub issues and pull requests comments [21]. The dataset is manually annotated with six distinct emotion classes: *Anger*, *Love*, *Fear*, *Joy*, *Sadness*, and *Surprise*. Among the comments, 17% convey *Anger*, 11% *Love*, 9.90% *Fear*, 21.10% *Joy*, 13.70% *Sadness*, and 16.40% *Surprise*. The remaining comments remain devoid of associated emotions.

Stack Overflow Dataset. Novielli et al. annotated a rich multi-label dataset comprising 4800 Stack Overflow questions, answers, and comments [22]. Within this dataset, 18.1% of the samples are labeled with *Anger*, 25.4% with *Love*, 2.2% with *Fear*, 10.2% with *Joy*, 4.8% with *Sadness*, and 0.9% with *Surprise*. The remaining contents of the dataset are neutral.

JIRA Dataset. Ortu et al. annotated a comprehensive collection of 4000 comments extracted from JIRA, classifying them into four distinct emotional categories: *Love*, *Joy*, *Anger*, and *Sadness* (1000 comments each) [2]. Within each category, *Love*, *Joy*, *Sadness*, and *Anger* account for 16.6%, 12.4%, 32.4%, and 30.2% respectively, while the remaining comments are neutral.

For training and testing with each dataset we employ an 80%-20% stratified sampling approach.

2.2 Emotion Model

All these three datasets rely on the well-known Shaver's tree-structured emotion model [23]. In Shaver's model, for each of the six basic emotions, there are secondary and tertiary-level emotions, which refine the granularity of the previous level. GoEmotions is an alternative emotion model used in the literature that was proposed by researchers at Google with the focus on emotions that can be observed in written text [24]. In their recent work, Imran et al. [21] extended Shaver's model by incorporating a few emotions from GoEmotions's [24] taxonomy in order to study emotions present in GitHub communications. Out of 27 emotions in GoEmotions' list, 26 are in the extended model by Imran et al. The only emotion that is not on the list is *Gratitude*.

In order to study both GoEmotions and Shaver's models, in this paper, we map the remaining emotion - *Gratitude* - within Shaver's tree-structured emotion model. We look into the definitions - how the authors defined *Gratitude* in GoEmotions [24] and if any emotion is defined similar way in Shaver et al. [23]'s definition. GoEmotions defined *Gratitude* as *"a feeling of thankfulness and appreciation."*, while Shaver et al. defined *Love "involving the appreciation of someone."* Therefore, we mapped *Gratitude* as a secondary emotion to the basic emotion *Love* in this study.

The extended model is shown in Table 1, with blue-colored emotions also appearing in the GoEmotions' listing.

2.3 Compared Models

Existing SE-specific models. We use three existing SE-specific models that have been shown to produce state-of-the-art performance in emotion classification.

- *ESEM-E*: A tool that is proposed by Murgia et al. [25] that uses unigram and bigram as features and an SVM as the ML model. It has been widely used in the literature for software engineering emotion classification tasks [21, 25, 26].
- *EMTk*: EMTk is proposed by Calefato et al. [27]. EMTk uses unigram, bigram, emotion lexicon, politeness, and mood as features and SVM as the ML model. Similar to ESEM-E, it has been widely used in the SE community [21, 22, 26].
- *SEntiMoji*: This deep learning-based model is proposed by Chen et al. [26], and built on top of the DeepMoji [28] model. This flexible model can identify different emotion categorization schemes, including Shaver's categorization.

For EMTk and SEntiMoji, the authors published the model implementations. We use the provided code for training and testing. As for ESEM-E, we carefully read the instructions provided by the authors and implemented the model by ourselves.

Table 1: Extended Shaver's tree-structured taxonomy.

Basic Emotion	Secondary Emotion → Tertiary Emotion
Anger 😠	Irritation → Annoyance, Agitation, Grumpiness, Aggravation, Grouchiness Exasperation → Frustration Rage → Anger, Fury, Hate, Dislike, Resentment, Outrage, Wrath, Hostility, Bitterness, Ferocity, Loathing, Scorn, Spite, Vengefulness Envy → Jealousy Disgust → Revulsion, Contempt, Loathing Torment Disapproval †
Love ❤️	Affection → Liking, Caring, Compassion, Fondness, Affection, Love, Attraction, Tenderness, Sentimentality, Adoration Lust → Desire, Passion, Infatuation Longing Gratitude ‡
Fear 😨	Horror → Alarm, Fright, Panic, Terror, Fear, Hysteria, Shock, Mortification Nervousness → Anxiety, Distress, Worry, Uneasiness, Tenseness, Apprehension, Dread
Joy 😀	Cheerfulness → Happiness, Amusement, Satisfaction, Bliss, Gaiety, Glee, Jolliness, Joviality, Joy, Delight, Enjoyment, Gladness, Jubilation, Elation, Ecstasy, Euphoria Zest → Enthusiasm, Excitement, Thrill, Zeal, Exhilaration Contentment → Pleasure Optimism → Eagerness, Hope Pride → Triumph Enthrallment → Enthrallment, Rapture Relief Approval † Admiration †
Sadness 😟	Suffering → Hurt, Anguish, Agony Sadness → Depression, Sorrow, Despair, Gloom, Hopelessness, Glumness, Unhappiness, Grief, Woe, Misery, Melancholy Disappointment → Displeasure, Dismay Shame → Guilt, Regret, Remorse Neglect → Embarrassment, Insecurity, Insult, Rejection, Alienation, Isolation, Loneliness, Homesickness, Defeat, Dejection, Humiliation Sympathy → Pity
Surprise 😲	Surprise → Amazement, Astonishment Confusion † Curiosity † Realization †

Notes: Emotions in blue appear in the list of emotions proposed by GoEmotions. Emotions added by Imran et al. from GoEmotions' list onto Shaver's taxonomy are denoted with †. A single emotion – *Gratitude* – is added to the taxonomy by this paper, denoted by ‡.

Fine-tuned LLMs. We fine-tune two popular LLMs – BERT and RoBERTa – that have been widely used as emotion and sentiment analysis, including in software engineering [29–32]. We leverage the pre-trained model weights from HuggingFace [33].

- *BERT*: Bidirectional Encoder Representations from Transformers is a widely-used LLM developed by Google. BERT is pre-trained using English Wikipedia and BooksCorpus [34].
- *RoBERTa*: Robustly Optimized BERT, a variant of BERT, is developed by Meta. It is pre-trained using English Wikipedia, BooksCorpus, news articles, Web text, and stories [35].

Zero-shot LLMs. We use three (two commercial and one open-source) recent pre-trained and instruction-tuned models in a zero-shot setting, i.e., the models are not tuned for the task of emotion (cause) detection in software engineering.

- *ChatGPT (GPT-3.5-turbo)* [13]: We use the gpt-3.5-turbo API by OpenAI. GPT-3.5-based models are pre-trained on a massive corpus of text data from diverse sources, including books, articles, websites, and other publicly available online content. The model was then instruction-tuned (from a large dataset of instructions with desired output) using Reinforcement Learning from Human Feedback (RLHF) [36].
- *GPT-4* [14]: We use the gpt-4 API by OpenAI. GPT-4 is a transformer-style model pre-trained using both publicly available data and data licensed from third-party providers; details of the training data are not released at the time of writing. GPT-4 introduced a rule-based reward model (RBRM) approach on top of RLHF.
- *flan-alpaca* [19]: This is a variation of the Alpaca [37] fine-tuned model. Alpaca was developed by Stanford, based on Meta's LLaMA [38] model using 52K instruction-based data instances. Due to licensing issues, the original Alpaca model is not accessible at the time of our experiment. Instead, using the Alpaca instructions dataset, Chia et al. [19] fine-tuned Google's instruction-tuned Flan-T5 [39] model and released the weights. We use the flan-alpaca-xl version from Hugging Face[1].

2.4 Metrics

The F1-score is a widely used metric for assessing the effectiveness of a (multi-class) classification model. It is the weighted harmonic mean of precision and recall, which takes into account both false positives and false negatives. $F1 - score = 2 * \frac{Recall * Precision}{Recall + Precision}$. To calculate the average score across all classes, i.e., emotions, we use the micro-averaged variant which has been widely used in related tasks [21, 32, 40].

2.5 Basic Emotion Prompting

The zero-shot LLMs we are considering are all instruction- (or prompt-) tuned. This recent category of LLMs use a fine-tuning process with instructional data, which helps the LLMs to better comprehend and respond to user-composed prompts. To our knowledge, there is no prior work on how to formulate prompts for emotion recognition in software engineering text using these LLMs.

A recent study by Kocon et al. [41] evaluated the performance of ChatGPT on various natural language processing tasks by designing over 38k prompts that covered 25 different tasks, including emotion classification using the GoEmotions dataset[24]. Inspired by this study, we designed a prompt for emotion classification that we used on all three datasets. More specifically, we asked the models to act as a user in a specific platform, i.e., GitHub, Stack Overflow, and JIRA, and provided the utterances and a list of the basic (top-level)

[1]https://huggingface.co/declare-lab/flan-alpaca-xl

emotions: *Anger*, *Fear*, *Love*, *Joy*, *Sadness*, and, *Surprise*. The prompt is the following:

> You are a [GitHub/Stack Overflow/JIRA] user. You are reading comments from [GitHub/Stack Overflow/JIRA]. Your task is to detect whether there is one of the following emotions aroused in you while reading the utterance.
> Emotions List: Anger, Fear, Love, Joy, Sadness, Surprise.
> Utterance: *<insert utterance>*.
> If there is no emotion in the text, write Neutral. Otherwise write exactly one word, the exact emotion from the emotions list.

Since the JIRA dataset does not contain *Fear* and *Surprise*, we do not list these two emotions in the prompt when evaluating with this dataset.

Results and Discussion. Table 2 shows the results for the three emotion classification datasets and for all the models. It is clearly noticeable from the results that the zero-shot LLMs performed poorly across all datasets, lagging behind the SE-specific models and the fine-tuned LLM models. The fine-tuned LLMs performed best, e.g., RoBERTa achieved the best micro-averaged F1-score overall by averaging 0.592, 0.735, and 0.818 respectively for GitHub, Stack Overflow, and JIRA datasets. Among the SE-specific models, the deep learning-based SEntiMoji model performed best with an average F1-score of 0.529.

In order to understand where the zero-shot LLMs are making mistakes, next, we conduct an error analysis.

Error analysis. One of the most common errors we observed is that zero-shot LLMs are misclassifying *Love* utterances as *Joy* for all datasets. For example, on the Stack Overflow dataset, the F1-score for *Love* is 0.0, 0.116, and 0.078 for flan-alpaca, ChatGPT, and GPT-4 respectively. Compared to this, BERT, RoBERTa, ESEM-E, EMTk, and SEntiMoji obtained an F1-score of 0.840, 0.861, 0.757, 0.811, and 0.829 respectively. This is also evident in the number of false positive (FP) utterances in the *Joy* category, i.e., for the Stack Overflow dataset, the number of FPs for BERT, RoBERTa, ESEM-E, EMTk, and SEntiMoji are 34, 21, 29, 17, and 17 respectively, whereas, for flan-alpaca, ChatGPT, and GPT-4, the FPs are 259, 72, and 91.

Another common type of error was that the models predicted *Neutral* often. For instance, on the GitHub dataset GPT-4 identified 269 (67%) utterances as *Neutral*. In many cases a secondary or tertiary emotion for Shaver's categorization most closely describes the annotated utterances. However, those emotions were not provided to the model. For example, consider the following sentence from the GitHub dataset: *"Any updates on this? I'm implementing a flutter application with barcode scanners, the soft keyboard on screen is really annoying."*, annotated as *Anger* and, on a more granular level, as *Annoyance*. All zero-shot LLMs models predicted it as *Neutral*. As another example, the following sentence is annotated as *Worry*, which is a tertiary-level emotion of *Fear*: *"My concern is that more new atributes may appear [...] it may break their behavior."*, while flan-alpaca and ChatGPT classified it as *Neutral*.

We also observed a number of hallucinations in the zero-shot LLMs output [42], where the models generated responses that were outside of what was asked. This led to situations where the models outputted emotions such as *Apology* and *Appreciation*, despite them not being in the prompted emotions list. For example, GPT-4 predicted the following sentence as *Apology*: *"Doh. Sorry for wasting your time."* even though the set of basic emotions provided in the prompt does not contain this emotion.

In order to address these issues, we experiment with constructing prompts with a more granular level of emotions, i.e., by considering the second and tertiary-level emotions in Shaver's extended taxonomy. This is also motivated by the study of Kocon et al. [41], who used all of GoEmotions' 27 emotions in their prompting experiments with ChatGPT.

Table 2: Micro-averaged F1-score of emotion classification models for three different datasets.

	GitHub [21]	SO [22]	JIRA [2]
SE-Specific			
ESEM-E	0.440	0.674	0.744
EMTk	0.434	0.651	0.734
SEntiMoji	0.529	0.721	0.793
Fine-tuned			
BERT	0.588	0.716	0.817
RoBERTa	**0.592**	**0.735**	**0.818**
Zero-shot			
ChatGPT	0.234	0.339	0.276
flan-alpaca	0.424	0.293	0.432
GPT-4	0.355	0.444	0.256

2.6 Granular-level Emotion Prompting

In order to experiment with more granular emotions, we require a labeled dataset that includes these emotions. Therefore, we specifically conducted these experiments with Imran et al. [21]'s dataset, which provides a secondary and tertiary-level emotion annotation while the other datasets do not. First, we conducted prompt experiments using a part of Imran et al.'s training set (note that the zero-shot LLMs are not using the training data) varying the information used in the prompts for each instruct-tuned language model. More specifically, we randomly selected 400 comments from the training dataset using stratified sampling and tested with granular-level prompting using the following strategies: 1) all emotions (basic, secondary and tertiary) from the extended Shaver's categories – a total of 140 emotions; 2) only the basic and secondary emotions from the extended Shaver's categories – a total of 34 emotions; 3) GoEmotions' list of 27 emotions.

We mapped the output emotion from the secondary and tertiary emotions to corresponding basic emotions as shown in Table 1 and compared the results of the models at this level (as the SE-specific models can only produce results at the basic emotion level). We also found during the granular-level prompting that the models sometimes produced minor wording variations of the provided emotions, such as *Confused* instead of *Confusion*, *Excited* instead of *Excitement*. While mapping the outputs of the zero-shot LLMs to the basic emotions, we made adjustments as not to punish the models for these minor differences.

During our experimentation, we observed that all three models tend to suffer more strongly from the issue of hallucination [42] when the complete emotion list (all of basic, secondary and tertiary

Table 3: Micro averaged F1-score of emotion classification for different models using Imran et al.'s dataset. The zero-shot LLMs use the GoEmotions list of 27 emotions.

	Imran et al. (N=2000) [21]						
	Anger	Love	Fear	Joy	Sad.	Surprise	Micro Avg.
SE-Specific							
ESEM-E	0.309	0.644	0.291	0.378	0.524	0.500	0.440
EMTk	0.430	0.682	0.163	0.378	0.525	0.345	0.434
SEntiMoji	0.460	0.642	0.377	0.556	0.629	0.458	0.529
Fine-tuned LLMs							
BERT	0.506	**0.712**	0.536	**0.579**	**0.636**	0.594	0.588
RoBERTa	0.525	0.683	0.492	0.500	0.613	0.673	**0.592**
Zero-shot LLMs							
ChatGPT	0.337	0.492	0.182	0.458	0.417	0.511	0.429
flan-alpaca	0.447	0.537	0.140	0.446	0.451	**0.740**	0.506
GPT-4	0.409	0.698	0.049	0.446	0.487	0.524	0.482

level emotions) are provided. For example, GPT-4 suffered 50% more hallucinations when the complete list is provided compared to the basic list of emotions. In particular, the models tended to generate extrinsic hallucinations [43], i.e., information beyond what is asked in the given prompt. This led to situations where the models generated emotions such as *Concern*, *Apology*, and *Appreciation*, despite them not being in the prompted emotions. This suggests that providing a very large list of emotions may not be optimal.

Out of the strategies we attempted, providing GoEmotions' 27 emotions list produced the best performance. For example, on the sample of the training dataset, ChatGPT achieved an F1-score of 0.201 when all emotions from Table 1 were provided, 0.341 when basic and secondary emotions are provided, and 0.419 when GoEmotions' emotions are provided. As noted earlier that the GoEmotions' taxonomy is developed specifically for text-based emotion recognition [24]. This can explain why it performed better than emotions selected directly from Shaver's taxonomy, which was developed based on psychological evidence and not specifically for text [23].

Therefore, we opt to use GoEmotions' list of emotions for prompting for emotion classification using the zero-shot LLMs. Next, we report the results on the Imran et al. [21]'s held-out test dataset.

Results and Discussion. Table 3 shows the results for emotion classification on Imran et al.'s GitHub dataset for all the models. Overall, BERT and RoBERTa still achieve the best results with an average F1-score of 0.588 and 0.592 respectively, while among the SE-specific models deep learning-based SEntiMoji achieved the highest average F1-score of 0.529. From the table, it is clear that all three zero-shot LLMs improve in most categories of emotions and overall micro-averaged F1-score. It is also noticeable that they improved in distinguishing *Love* and *Joy* utterances. However, the zero-shot LLMs still perform badly for *Fear*. Overall, surprisingly, the open-source model flan-alpaca achieved the best performance with an average F1-score of 0.506 – an improvement of 19.33% from the basic emotion-level prompting, while the proprietary model GPT-4 achieved 0.482 – an improvement of 35.77%. Both of these are improvement over the three SE-specific models and the proprietary ChatGPT (*gpt-3.5-turbo*) model.

The results again point out that despite there having been major advancements in instruction-tuned LLMs, the fine-tuned deep learning models still perform better for specific, well-defined tasks that require domain-specific knowledge. To understand more where zero-shot LLMs are still making errors in detecting emotions, we conduct an error analysis on the errors in granular level prompting.

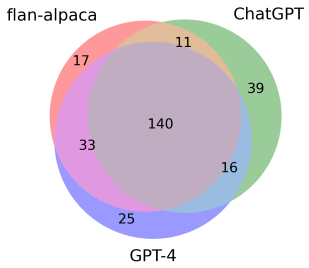

Figure 1: Wrongly classified utterances among the zero-shot LLMs.

Error Analysis. From the 356 non-*Neutral* instances in the test set, all three models correctly predicted 75 instances, two models made the right prediction for 67 instances, only one of the models made the right prediction for 74 instances, and no zero-shot LLMs made the right prediction on 140 instances. A Venn diagram of the classification error of each of the three zero-shot LLMs is shown in Figure 1. We examine more closely the 140 utterances where all three zero-shot LLMs models made wrong predictions.

As also noticeable in Table 3, *Fear* is the most often misclassified category with (35/140) instances. The errors in this category are especially discernible with GPT-4. With basic emotions only, GPT-4 achieved an F1-score of 0.353 in the *Fear* category while at the granular level the F1-score went down to 0.049. The primary reason for it is that GPT-4 generated hallucinated output with labels such as *Worry*, *Concern*, etc., which are missing in the GoEmotions list. However, some of these emotions are present in Shaver's extended list and in Imran et al.'s annotation. In the annotated data, most of the *Fear* utterances are due to the tertiary-level emotion *Worry*. For example, the utterance *"Isn't this a breaking change? Can we get away with it?"* is annotated as *Worry* (3rd level of *Fear*) in the ground truth. Another example is the utterance: *"I guess my concern is that it sets a precedent where somebody could see it and think that it would be fine to use in 'core'."*

The second most misclassified emotion category is *Joy* with (33/140) instances. Many of these errors are because the models are predicting conservatively, i.e., predicting *Neutral* instead of a specific emotion. For example, *"Anyway, the syntax change is fine."* – this utterance is annotated as *Approval* (2nd level of *Joy*). Another example, *"[USER] can you assign this ticket to me, I can help in this."* – this utterance is annotated as *Enthusiasm* (2nd level of *Joy*). Also, notable here is that *Enthusiasm* is not in GoEmotion's emotion list. Another type of error among *Joy* category is that they are often misclassified as *Surprise*. For example, *"This was actually causing this test-case not to be executed!"* - this utterance is annotated as *Relief* (2nd level of *Joy*), but the flan-alpaca and GPT-4 model predicted as *Surprise*.

The third most misclassified category is *Sadness* with (31/140) instances. We observed that these utterances are often misclassified as *Anger*, *Surprise*, or *Neutral*. For example, flan-alpaca and GPT-4 predicted *Surprise* for this utterance: *"Ah sorry I thought 'ScaleUpdateDetails' was constructed in '_update' nvm."*

We observed hallucinated emotions as well (5 for GPT-4, 13 for flan-alpaca, 2 for ChatGPT), especially *Concern* and *Worry* among *Fear* utterances; and *Appreciation* among *Love* utterances.

Overall, the error analysis points out the need for having a more specialized emotion taxonomy for text-based emotion detection, in particular for software-engineering-related text. As noted earlier, Shaver's [23] taxonomy, developed in Psychology, includes many additional emotions that do not appear in the text and confuse the zero-shot LLMs. Meantime, while GoEmotions list focuses on text-based emotions, they are still missing some commonly observed emotions in software engineering such as *Worry* and *Frustration*.

3 EMOTION-CAUSE EXTRACTION

The results of the preliminary study suggest that zero-shot LLMs are capable at detecting emotion categories when provided with granular level emotions, performing slightly worse than the best evaluated models (i.e., in Table 3, flan-alpaca's F1-score is 0.506 relative to SEntiMoji's 0.529 and RoBERTa's 0.592). In this section, we examine their feasibility for the more challenging task of emotion-cause extraction.

The use of LLMs for emotion-cause extraction has experienced a notable uptick in interest in recent years [44, 45]. Emotion-cause extraction seeks to identify the cause or event that instigates a specific emotion in a given text, providing essential insights into human behavior and deepening our comprehension of the underlying emotions behind text-based communication. Researchers have explored the potential of LLMs in detecting emotion causes across multiple domains, such as social media and news articles [44–46].

Despite the growing interest in emotion-cause extraction in different domains, there is a lack of research on this problem in software engineering communication text. This research gap inspires our study, which aims to investigate the effectiveness of zero-shot LLMs in detecting emotion causes in GitHub comments.

To this end, we first manually annotate emotion causes in a subset of Imran et al.'s [21] data, identifying the text span that represents the cause of emotion in the comment. We then use zero-shot LLMs to extract emotion causes and compare their performance against the annotated emotion causes using the BLEU score [47], a standard metric in machine translation to evaluate text sequence similarity. Below, we present a detailed description of our annotation process, zero-shot LLMs, and the comparison of BLEU scores across different models and configurations.

3.1 Annotation

To create a dataset for the emotion-cause extraction task, we begin by selecting 75 utterances for each of the 6 basic emotion categories (*Anger, Love, Fear, Joy, Sadness, Surprise*) from Imran et al.'s training dataset, totaling 450 utterances. Two senior undergraduate students (with 3+ years of experience in programming) are then tasked with annotating the dataset by identifying emotion causes, if any, based on the previously annotated basic, secondary, and tertiary emotions by Imran et al. We provide them with the following instructions:

> *For each instance containing an emotion (Anger, Love, Fear, Joy, Sadness, Surprise), find the span of text (if any) that contributes to the annotated emotion. Each instance then should be annotated with its corresponding causes if existing. Emotion can sometimes be associated with more than one cause, in such a case, both causes should be marked. Since in some cases, more than one emotion can be present in an instance, the causes for emotion should be mapped as <emotion, cause span>.*

The above instructions are adapted from Chen et al.'s seminal work on detecting emotion causes [48]. We also provide the annotators with definitions and examples of different types of emotion causes. After the annotation task is completed, one of the authors of the paper manually reviewed both sets of annotations and noted disagreements in 44 of the 450 instances. To resolve these discrepancies, the annotators are asked to meet on Zoom and discuss and resolve their differences. This process ensures the annotated dataset's reliability and consistency.

3.2 Model Selection

For the automated emotion-cause extraction task, we evaluate the same three instruction-tuned models (ChatGPT, GPT-4, and flan-alpaca) that we used for emotion detection in Section 2, i.e., the preliminary study. We do not use BERT or RoBERTa as those models require a large amount of domain-specific training data [49], which we lack.

3.3 Prompt Design

The structure of our emotion-cause extraction prompt is intended to mimic a real-world scenario where a GitHub user is going through issues and pull requests, experiencing various emotions, and trying to pinpoint the cause of a specific emotion in a given utterance. We use a two-step prompt that asks the model to first detect the emotion in the utterance using the procedure outlined in Section 2. Then, we prompt the model to identify the cause of this emotion, as shown in the framed box structure.

> You are a GitHub user. You are reading GitHub comments. Your task is to extract the span that is causing the emotion *<insert emotion>* in the following GitHub utterance: *<insert utterance>*.
>
> Write the span of the cause within a double quote.
>
> Do not write anything else.

3.4 Results

To ensure consistency in our evaluation, we preprocess all comments, annotated causes, and model-extracted causes by removing punctuation, lemmatizing, and stemming. After preprocessing, the average length of the 450 utterances is 28.08 words, while the average length of the manually annotated emotion cause spans is 7.43 words. We find that the emotion cause spans extracted by GPT-4, ChatGPT, and flan-alpaca have average lengths of 8.85, 8.64, and 13.12 words, respectively.

3.4.1 BLEU score. The BLEU (Bilingual Evaluation Understudy) score is a metric used to evaluate the quality of machine-generated text by comparing it to human-generated reference text [47]. The BLEU score measures the similarity between the machine-generated text and the reference text based on the n-gram overlap between

Table 4: BLEU scores of different zero-shot LLMs.

Model	BLEU-1	*BLEU-2*	BLEU-3	BLEU-4
ChatGPT	0.522	*0.489*	0.467	0.450
GPT-4	**0.637**	***0.598***	**0.571**	**0.554**
flan-alpaca	0.571	*0.543*	0.525	0.508

them. The higher the BLEU score, the closer the machine-generated text is to the reference text. The formula for the BLEU score is: $BLEU = BP \cdot \exp\left(\sum_{n=1}^{N} w_n \cdot \log(p_n)\right)$. where,

- BP is the brevity penalty, which is 1 if the machine-generated text is longer than the reference texts and less than or equal to them otherwise.
- N is the maximum n-gram order.
- p_n is the precision score for n-grams.
- w_n is the weight for n-grams, which is usually set to $\frac{1}{N}$ for uniform weighting of all n-gram orders.

3.4.2 BLEU Score Interpretation. The interpretation of BLEU scores can vary depending on the specific domain and language being evaluated. In the software engineering domain, a BLEU score is commonly used to evaluate the quality of generated bug reports, code comments, and code summarization. Denkowski and Lavie [50] suggest that BLEU scores above 0.30 generally indicate that the generated text is understandable, while scores above 0.50 are indicative of good and fluent results. Previous research [50, 51], including studies in software engineering [52], has used this scale to interpret the results of BLEU scores. It is important to note that the choice of n-gram order used to calculate the BLEU score can impact the final score; typically, 4-gram is used for BLEU score calculation [47, 50]. In the case of our study, however, the emotion cause spans are often short, making the bigram a more suitable choice for BLEU score calculation, i.e., BLEU-2.

3.4.3 Discussion. The BLEU scores for the three models using unigram, bigram, trigram, and four-gram are shown in Table 4. The score ranges between 0.450 to 0.637, which indicates that all models are generally able to extract the right emotion causes to some extent, especially GPT-4 and flan-alpaca as both models' BLEU scores are always above 0.5.

When considering BLEU-2, GPT-4 obtains the highest score of 0.598, followed by flan-alpaca with 0.543 and ChatGPT with 0.489. Out of 450 utterances, 107 cases are identified where all three models' BLEU-2 scores are higher than 0.5. We observe that these 107 utterances are relatively short, with an average length of 15.26 words, while the annotated cause spans have an average length of 7.02 words. The three models, GPT-4, ChatGPT, and flan-alpaca, extract similar length spans on average, which are 7.89, 7.79, and 7.95 words, respectively. For example, in the following utterance, *"I'm not sure how to fix this, nor if this is acceptable in this test case. Namespaces in TS are magic to me 😅"*, the annotated cause of *Amusement* (3rd level *Joy*) is *"Namespaces in TS are magic to me"*. GPT-4 also extracted the same span as the cause. However, it is not always the case that the annotated cause span completely overlaps with the spans extracted by the models. For example, in this utterance, *"Oh, you didn't add composes and values. Well, I like it even more. Those features are hard to maintain."*, the annotated cause span is *"I like it even more"*, and the extracted cause span by GPT-4 is *"Well, I like it even more."*

Out of 450 utterances, we observe that in 41 cases, all three models' BLEU scores are less than 0.30. These comments are relatively longer, containing an average of 44.17 words, while the annotated cause spans contain an average of 5.05 words. The extracted average lengths of spans for GPT-4, ChatGPT, and flan-alpaca are 10.10, 13.14, and 22.83 words, respectively.

3.4.4 Error Analysis. To gain insight into the models' mistakes, we analyze the 41 utterances where all three models had a BLEU score of less than 0.30. Our examination reveals that the errors can be classified into a few primary categories, which are elaborated below.

Incorrect Emotion. The main source of error for all three models is the misidentification of the emotion expressed in the utterance (24/41 utterances). This misidentification leads to the detection of an incorrect cause event. For instance, consider the utterance, "*Oh right! 🙃 This started as a Mac issue, I forgot to add the rest.*" The annotated emotion for this utterance was *Neglect* (2nd level *Sadness*) and the annotated cause span is "*I forgot to add the rest.*" However, ChatGPT identifies the utterance as *Confusion* (2nd level *Surprise*) and extracts "🙃" as the cause event instead. GPT-4 detects *Amusement* in the utterance and extracts the cause span as *"Oh right! 🙃."* Meanwhile, flan-alpaca identifies "*Curiosity* (2nd level *Surprise*)" and extracts the cause span as *"Oh right!"* This error category emphasizes the importance of accurately detecting the emotion expressed in the text before extracting emotion causes.

Incorrect Cause. This error occurs when the models correctly classify the emotion but detect a different cause than the ground truth (12/41 utterances). For example, in the following utterance *"[USER] yep, it is bug, we will fix it, so we have it in 'experiments' :+1:"*, the annotated emotion is *Approval*, and the annotated cause span is *"it is a bug"*, while GPT-4 detected the cause span *"we will fix it"*. This error category highlights the difficulty in identifying the exact cause of events in conversational text, especially in longer, multi-part comments.

Hallucinations. In addition to the two error categories described above, we also observe instances of hallucinations in the cause event extraction process. In some cases, the models' output *"the entire sentence."*, *"the span: <followed by the span>"*, *"span starting from word X to word Y"*, and other nonsensical outputs. We observe that ChatGPT produces more hallucinated data than the other two models, which is one reason why its BLEU score is lower. This highlights the need for continued research into developing more accurate and reliable models that can follow the prompt exactly.

4 INVESTIGATING THE CAUSES OF FRUSTRATION IN THE TENSORFLOW REPOSITORY: INSIGHTS FOR PROJECT MAINTAINERS

We perform a case study on how understanding emotion-cause can be helpful in real-world scenario. *Frustration* is a pervasive emotion in software development [20], and it is particularly relevant in the context of open-source projects [53]. Wrobel et al. noted

that *Frustration* is the most commonly felt emotion during software development [54]. Collaborative work, lack of control over external contributors' code, and the complexity of software development processes can all contribute to the *Frustration* of developers and end-users. In contrast to other emotions, such as *Confusion* or *Excitement*, *Frustration* is more strongly associated with obstacles, challenges, and difficulties. It is also often accompanied by other negative emotions, such as *Anger*, *Disappointment*, or *Helplessness* [55]. Given the complexity and collaborative nature of open-source software development, *Frustration* is undesirable but likely to be a common experience for many contributors and users. Therefore, understanding the causes of *Frustration* in open-source development can provide valuable insights for project maintainers into what are the key issues that impede collaboration and the productivity of project participants.

Tensorflow[2] is a popular open-source platform for developing machine learning models and has a large number of developers and a huge user-base, which makes it an interesting case study for investigating the causes of *Frustration* in open-source software development. For instance, monitoring of the causes of *Frustration* in TensorFlow contributors can aid in the construction of project maintainer dashboards that help attract and retain open source contributors [56, 57].

4.1 Data Collection and Cause Extraction

To conduct our analysis, we collect all publicly available issues and pull requests comments made on the Tensorflow repository, hosted on GitHub, between March 30, 2022, and March 30, 2023. We choose this time period to ensure that our analysis covers a recent and substantial range of comments. Most GitHub repositories, including Tensorflow, differentiate different types of comment authors based on their relationship to the project, such as Contributors, Collaborators, Members and None[3]. A Collaborator is a GitHub user invited to work on the repository, a Contributor has committed code before, a Member belongs to the owning organization, and None has no affiliation with the repository. Collaborators, Contributors, and Members are active developers, while None comprises user commenters. To analyze software developer *Frustration*, we exclude comments from the None category.

Following the emotion-cause extraction procedure described in Section 3, we extract the emotions and causes of each comment. We use the *flan-alpaca* model for this purpose, as it performed reasonably well in both emotion detection and emotion-cause extraction tasks compared to the proprietary zero-shot LLMs. Another advantage of *flan-alpaca* is that it is open-source and its weights are publicly available. This ensures the reproducibility of our results. In contrast, closed-source LLMs may become unavailable, e.g., OpenAI's Codex LLM was deprecated in March, 2023.

We collect only the utterances that the model identified as expressing *Frustration*, resulting in a dataset of 1275 comments.

[2]https://github.com/tensorflow/tensorflow
[3]https://docs.github.com/en/graphql/reference/enums#commentauthorassociation

4.2 Clustering

To identify common themes among the causes of *Frustration*, we employ the DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm [58]. It has been effectively used in previous software engineering studies involving clustering textual data [59, 60]. The main advantage of using the DBSCAN algorithm is that it does not require a pre-specified number of clusters, which can be difficult to estimate in advance. This is particularly useful in the context of identifying common themes among the causes of *Frustration*, as it is difficult to know beforehand what the common themes are. Another advantage of the DBSCAN algorithm is its ability to automatically handle noise and outliers, which is relevant as the extracted causes by *flan-alpaca* can contain errors, as discussed in the previous sections.

While DBSCAN does not require to specify the number of clusters, it requires two key parameters [61]: 1) ϵ - a real positive value - the maximum allowed distance between two samples to be considered that they are part of the same dense region, and 2) $MinPts$ - a small positive constant integer - the minimum number of samples required to consider a dense region as a cluster. We performed a manual parameter sweep, testing ϵ values from 0.1 to 0.8 in increments of 0.05, and $MinPts$ values from 2 to 6, following standard guidelines for parameter tuning in machine learning and data mining [62]. Based on the number of clusters, average number of elements per cluster, and cluster composition, we selected ϵ = 0.3 and $MinPts$ = 4, which yielded 23 clusters. Before applying the DBSCAN algorithm, we perform standard text pre-processing such as removing punctuation, URL removal, and lemmatizing on the list of causes. We use the scikit-learn library's implementation of the DBSCAN algorithm with cosine similarity and sentence-level embeddings (*all-mpnet-base-v2* model [63]).

To focus our analysis on the most common causes of *Frustration*, we limit our discussion to the top 6 clusters in terms of the number of comments in each cluster. The clusters are presented in Table 5, along with their description, size, and examples. We read the GitHub comments and the emotion causes to identify the underlying theme in each cluster that leads to *Frustration*.

4.3 Causes of Frustration

We utilized *thematic analysis* to identify the themes of the clusters [64]. Specifically, one of the authors of this paper read each comment and coded the initial themes. Then another author reviewed the themes, then both authors discussed resolving discrepancies and finalizing the themes until the analysis reached saturation, with no new themes emerging [65]. Each cluster theme is described below:

TensorFlow Version and Dependency Issues: This cluster primarily includes project participants struggling with incompatibility issues due to version mismatches between TensorFlow and its related dependencies. They express frustration over difficulties in configuring TensorFlow to operate correctly on their system. They also express frustration over transitioning from legacy versions to newer versions. One possible way to address these issues is to provide a more comprehensive documentation on version compatibility between TensorFlow and its dependencies.

Table 5: Clusters of causes of *Frustration* in TensorFlow project participants in GitHub.

	Cluster Description	Count	Example Comments
1	**TensorFlow Version and Dependency Issues**: This cluster focuses on build and compatibility problems across various TensorFlow versions, challenges in reproducing issues in specific TensorFlow versions, and complications with related libraries and plugins such as TensorRT and Keras.	58	(1) *[USER] Your original issue looks like* ***you have a bad version of tensorflow_io_gcs_filesystem installed****. [...].* (2) ***It's probably not a bug in Tensorflow but Apple's tensorflow metal plugin****. See for example the following discussion [...]*
2	**Pull Request Delays and Merge Conflicts**: The cluster comprises developer frustration from unresolved merge conflicts and from delays in merging pull requests.	26	(1) *[...] But* ***there are a bunch of merge conflicts****. Since Random seeds are such a common topic in software [...]* (2) *It might have been a* ***wrong-way merge*** *or something like that. At this point it's usually easier to just close it [...]*
3	**Failing Tests**: This type of *Frustration* arises from the ambiguity and complexity of test failures, which make it challenging for project participants to determine whether the issues are linked to their code changes or are caused by unrelated factors.	15	(1) *[USER]: It is just a first draft.* ***The test doesn't even work****. In the meantime, [...]* (2) *[...] Yes, I'll work on this.* ***It's weird that these tests are failing because I thought I ran them successfully for PR*** *[...]*
4	**Too Fine-Grained Commits**: The cluster reflects developer *Frustration* caused by too granular commits in the repository. Some developers request a commit history devoid of incremental commits that represent only partial progress on a change task.	9	(1) *Can you squash these commits please?* ***It doesn't make sense to have 5 commits for a line change*** *and one extra empty line.* (2) ***3 commits for a single line change****? Can you please merge the commits in just one? [...]*
5	**CI Flakiness**: This type of *Frustration* is caused by Continuous Integration (CI) failures that seem unrelated, inconsistent, or uninformative to developers.	8	(1) *[USER]* ***there was failed ci.*** *Is there anything to do?* (2) ***CI failure does not look related to these changes****, seeing the same failure on #56345 [...] so I assume this is noise. [...]*
6	**CUDA/CuDNN Compatibility Issues**: This cluster reflects the *Frustration* experienced when dealing with compatibility issues related to CUDA and CuDNN.	8	(1) *Unfortunately this change needs to be rolled back,* ***it seems it breaks JAX build under CUDA 11.4 and CuDNN 8.2*** (2) *[...] - Did you downgrade the CUDA to 11.2?* ***Looking at Nvidia docs it looks like the display driver and cuda driver do not match*** *[...]*

Pull Request (PR) Delays and Merge Conflicts: This cluster is related to PR merging and associated communication, as well as merge conflicts. The project participants express *Frustration* when they have to wait a long time for a PR to be reviewed or merged by the project maintainers. Merge conflict-induced *Frustration* is a well-known issue in open source software development [66]. Implementing automated review bots and streamlined conflict-resolution procedures can help mitigate this form of *Frustration*.

Failing Tests: The cluster highlights the *Frustration* felt due to test failing, possibly flaky tests [67]. The project participants report two main sources of *Frustration*: first, the inability to identify the root cause of test failures that seem unrelated to their code changes; second, unexpected test failures leading to their PRs being reverted.

Too Fine-Grained Commits: This cluster reflects developers' *Frustration* on commits that capture incomplete changes or partial progress on a task, which need to be squashed. The comments demonstrate developer sensitivities around balancing incremental changes with maintaining a coherent commit history. Setting PR guidelines about git commit hygiene can help to mitigate this issue.

CI Flakiness: Like test flakiness, CI flakiness is another common source of developer *Frustration* [67, 68]. This cluster highlights the complexity of CI failures. The *Frustration* is evident as the developers grapple with failed CI tests, yet believe these problems are unrelated to their own contributions.

CUDA/CuDNN Compatibility Issues: The project participants express *Frustration* regarding GPU library compatibility. This reflects the challenge of managing interdependent, rapidly evolving software ecosystems [69]. TensorFlow relies on quickly changing GPU libraries like CUDA and CuDNN. Expanding CI testing across more diverse versions, detecting CUDA/CuDNN versions and alerting if incompatible, and explicitly documenting supported versions can help to reduce this pitfall.

5 RELATED WORK

The related work can be divided into three parts: prompt engineering for zero-shot LLMs, automated emotion-cause extraction in NLP, and the role of emotions in software engineering.

Prompt Engineering for Zero-Shot LLMs. Zero-shot learning, a task where a model is trained to recognize and classify unseen classes without any explicit training data for those classes, has been a recent focus among researchers and practitioners for a variety of tasks, including image and text classification, question answering, language generation, and data augmentation [49, 70–72]. Recently, researchers have focused on leveraging LLMs for zero-shot learning [73–76]. In the context of zero-shot learning, prompt engineering with LLMs has emerged as an area of interest in recent years [18, 74, 75, 77]. One approach that has been explored is the use of task-specific prompts, which are designed to elicit the desired response from the model. These prompts can be constructed manually or generated automatically and can be tailored to the specific task at hand [18]. For example, Brown et al. used an LLM to perform text classification using task-specific prompts [78]. Another approach is the use of general-purpose prompts, which are designed to be broadly applicable across a range of tasks [14, 75]. The recent advancements in language models such as ChatGPT [13], GPT-4 [14], BARD [79], LLaMA [38], and Alpaca [37] have made the general-purpose prompt approach increasingly popular. These models have achieved impressive performance across a range of tasks and continue to push the boundaries of NLP.

Automated Emotion-Cause Extraction in NLP. Automatically extracting emotion-cause has gained attention in recent years in NLP [44–46, 80–82]. Emotion-cause extraction is challenging, as both emotions and their causes can be expressed in various ways, including but not limited to explicit statements, implicit suggestions, and contextual cues. Several techniques have been proposed to address this challenge, including rule-based approaches, machine learning-based approaches, deep learning-based approaches, and LLM approaches [44, 83–85]. In recent years, the focus has been on LLM approaches [44–46]. Researchers have explored this area with prompting as well [86]. Wang et al. noted that ChatGPT achieves comparable performance on the emotion-cause extraction task in news articles [87]. In this study, we apply prompt-based emotion-cause extraction for three state-of-the-art LLMs, namely ChatGPT, GPT-4, and flan-alpaca [13, 14, 39].

The Role of Emotions in Software Engineering. Emotions play a crucial role in software engineering, as software development is a complex and collaborative process that often involves multiple stakeholders with different perspectives and priorities [4, 6, 88–90]. Researchers have explored the role of emotions in software engineering through qualitative analyses, quantitative analyses, and surveys [4, 6, 25–27, 54, 91–95]. Gachechiladze et al. conducted a study on where *Anger* is directed, i.e., towards self, others, and objects [7]. Ford et al. conducted a survey with 256 software developers to identify common sources of *Frustration* [20]. Graziotin et al. investigated the causes of unhappiness among software developers, using a survey of 2,220 participants [93]. Later, Graziotin et al. conducted a study of the effects of unhappiness [94]. More recently, there has also been a focus on studying conflicts, toxicity, and incivility in open source communities [90, 96–99].

To our best knowledge, there has been no research on the automated detection of emotion-causes in software engineering. To fill this gap, in this study, we examine the efficacy of existing state-of-the-art large language models in automatically extracting emotion-causes. We also perform a case study to demonstrate how these models can be applied in real-world scenarios.

6 THREATS TO VALIDITY

In this section, we discuss the potential threats to the validity of our study grouped into three categories: construct validity, internal validity, and external validity.

Construct validity. Construct validity is the extent to which our study accurately measures the concepts and constructs it aims to measure. One potential threat to construct validity is the use of automated zero-shot LLMs to extract emotion causes from domain-specific comments. These models are designed to perform general-purpose tasks and are not fine-tuned to extract emotion causes in software engineering communication text. To address this threat, we perform multiple error analyses to understand where these models make mistakes. Additionally, there could be a threat in the construction of the prompts. To mitigate this threat we followed existing literature and validated various versions of the prompt with labeled data in order to find a suitable prompt for the zero-shot LLMs. Another threat to construct validity comes from our manual labeling of the causes, which may introduce some subjectivity and bias, potentially impacting the accuracy of the reported results. We reduced this threat via multiple annotators and by resolving discrepancies to achieve 100% agreements.

Internal validity. The concept of internal validity relates to the degree to which the manipulation of an independent variable is responsible for the outcomes of a study. In our examination of an open-source project, *Frustration* causes represent an independent variable. However, there are potential threats to internal validity, such as unaccounted factors like prior experience with the project or technical expertise that could contribute to software developers' *Frustration*. Moreover, the use of flan-alpaca for extracting frustration causes could result in the misclassification of some utterances, leading to the potential omission of certain clusters that could provide alternative explanations for *Frustration* or identification of some clusters that do not in fact represent this emotion. Nevertheless, the use of DBSCAN reduces the effect of random noise, and the list of *Frustration* causes provided in Table 5 follow the software engineering literature on common problems developers face during open-source software development [100–102].

External validity. External validity pertains to the generalization of our study's findings to other settings and contexts. For emotion detection, we used the categories from extended Shaver's taxonomy as well as GoEmotions' taxonomy from previous research [21, 23, 24]. However, our findings may not necessarily be transferable to other emotion categories. Another potential threat to external validity is the specific nature of the open-source project we studied, i.e., TensorFlow. The project's characteristics, such as its size, development stage, and community culture, may not be representative of other open-source projects. Additionally, the programming language and technology stack used in the project may have influenced the types of causes of *Frustration* observed. Therefore, it is important to interpret our findings in the context of the specific project we studied and exercise caution when generalizing them to other open-source projects. Further investigation is needed to generalize these results beyond the three specific models and the data and projects we have used in our study.

7 CONCLUSIONS

In this paper, we presented an approach for automated emotion-cause extraction in software developer communication using three zero-shot LLMs, namely ChatGPT, GPT-4, and (the open-source) flan-alpaca, through a prompting approach. We first conducted a preliminary study to evaluate the models' performance in emotion classification tasks on an existing recent dataset, and we found that they perform well compared to state-of-the-art models. We then showed the feasibility of using these models for emotion-cause extraction on a subset of 450 utterances from the same dataset by manually annotating the emotion causes of these utterances and automatically extracting the causes using prompts. We compared the BLEU score performances of the models and found that GPT-4 achieved the highest BLEU-2 score of 0.598, followed by flan-alpaca with 0.543, and ChatGPT with 0.489. To demonstrate the possible real-world applications of emotion-cause extraction, we conducted a case study on the causes of *Frustration* in a large GitHub open-source project – Tensorflow.

There are several avenues for future work. First, our case study only focused on one emotion and one open-source project. Future

studies that use emotion-cause extraction should investigate other emotions and a broader range of projects to generalize our findings. Second, further work is needed to improve the accuracy of emotion-cause extraction from text in software engineering communication. This could involve few-shot prompting, fine-tuning language models, or developing domain-specific models tailored for software engineering communication. Overall, our study provides a starting point for future research to explore the potential of emotion-cause extraction in software engineering communication.

REFERENCES

[1] G. Destefanis, M. Ortu, S. Counsell, S. Swift, M. Marchesi, and R. Tonelli, "Software development: do good manners matter?" *PeerJ Comput. Sci.*, vol. 2, p. e73, 2016.

[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 *Proceedings of the 12th Working Conference on Mining Software Repositories*, ser. MSR '15. IEEE Press, 2015, pp. 303–313.

[3] R. Souza and B. Silva, "Sentiment analysis of travis ci builds," in *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*. IEEE, 2017, pp. 459–462.

[4] A. Murgia, P. Tourani, B. Adams, and M. Ortu, "Do developers feel emotions? an exploratory analysis of emotions in software artifacts," in *MSR 2014*, 2014.

[5] 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 *Product-Focused Software Process Improvement: 14th International Conference, PROFES 2013, Paphos, Cyprus, June 12-14, 2013. Proceedings 14*. Springer, 2013.

[6] K. Madampe, R. Hoda, and J. Grundy, "A framework for emotion-oriented requirements change handling in agile software engineering," *IEEE Transactions on Software Engineering*, 2023.

[7] D. Gachechiladze, F. Lanubile, N. Novielli, and A. Serebrenik, "Anger and its direction in collaborative software development," in *2017 IEEE/ACM 39th International Conference on Software Engineering: New Ideas and Emerging Technologies Results Track (ICSE-NIER)*. IEEE, 2017, pp. 11–14.

[8] G. Fucci, N. Cassee, F. Zampetti, N. Novielli, A. Serebrenik, and M. Di Penta, "Waiting around or job half-done? sentiment in self-admitted technical debt," in *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*. IEEE, 2021, pp. 403–414.

[9] F. Ebert, F. Castor, N. Novielli, and A. Serebrenik, "An exploratory study on confusion in code reviews," *Empirical Software Engineering*, vol. 26, pp. 1–48, 2021.

[10] P. Chatterjee, K. Damevski, and L. Pollock, "Automatic extraction of opinion-based Q&A from online developer chats," in *Proceedings of the 2021 IEEE/ACM 43rd ICSE*, 2021.

[11] X. Yu, W. Rong, Z. Zhang, Y. Ouyang, and Z. Xiong, "Multiple level hierarchical network-based clause selection for emotion cause extraction," *IEEE Access*, vol. 7, pp. 9071–9079, 2019.

[12] B. Xu, H. Lin, Y. Lin, Y. Diao, L. Yang, and K. Xu, "Extracting emotion causes using learning to rank methods from an information retrieval perspective," *IEEE Access*, vol. 7, pp. 15 573–15 583, 2019.

[13] OpenAI, "Chatgpt," https://openai.com/blog/chatgpt, 2023.

[14] ——, "Gpt-4 technical report," *ArXiv*, vol. abs/2303.08774, 2023.

[15] S. Carta, A. Giuliani, L. Piano, A. S. Podda, L. Pompianu, and S. G. Tiddia, "Iterative zero-shot llm prompting for knowledge graph construction," *arXiv preprint arXiv:2307.01128*, 2023.

[16] H. Zhuang, Z. Qin, K. Hui, J. Wu, L. Yan, X. Wang, and M. Berdersky, "Beyond yes and no: Improving zero-shot llm rankers via scoring fine-grained relevance labels," *arXiv preprint arXiv:2310.14122*, 2023.

[17] J. Wang, Y. Liang, F. Meng, B. Zou, Z. Li, J. Qu, and J. Zhou, "Zero-shot cross-lingual summarization via large language models," in *Proceedings of the 4th New Frontiers in Summarization Workshop*, 2023, pp. 12–23.

[18] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large language models are zero-shot reasoners," *Advances in neural information processing systems*, 2022.

[19] Y. K. Chia, P. Hong, L. Bing, and S. Poria, "Instructeval: Towards holistic evaluation of instruction-tuned large language models," *arXiv preprint arXiv:2306.04757*, 2023.

[20] D. Ford and C. Parnin, "Exploring causes of frustration for software developers," in *2015 IEEE/ACM 8th International Workshop on Cooperative and Human Aspects of Software Engineering*. IEEE, 2015, pp. 47–50.

[21] M. M. Imran, Y. Jain, P. Chatterjee, and K. Damevski, "Data augmentation for improving emotion recognition in software engineering communication," in *37th IEEE/ACM International Conference on Automated Software Engineering*, 2022.

[22] N. Novielli, F. Calefato, and F. Lanubile, "A gold standard for emotion annotation in stack overflow," in *Proceedings of the 15th international conference on mining software repositories*, 2018, pp. 14–17.

[23] P. R. Shaver, J. C. Schwartz, D. Kirson, and C. O'Connor, "Emotion knowledge: further exploration of a prototype approach." *Journal of personality and social psychology*, vol. 52 6, pp. 1061–86, 1987.

[24] D. Demszky, D. Movshovitz-Attias, J. Ko, A. Cowen, G. Nemade, and S. Ravi, "Goemotions: A dataset of fine-grained emotions," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 4040–4054.

[25] A. Murgia, M. Ortu, P. Tourani, B. Adams, and S. Demeyer, "An exploratory qualitative and quantitative analysis of emotions in issue report comments of open source systems," *Empirical Software Engineering*, vol. 23, pp. 521–564, 2018.

[26] Z. Chen, Y. Cao, X. Lu, Q. Mei, and X. Liu, "Sentimoji: an emoji-powered learning approach for sentiment analysis in software engineering," in *Proceedings of the 2019 27th ACM joint meeting on ESEC/FSE*, 2019.

[27] F. Calefato, F. Lanubile, N. Novielli, and L. Quaranta, "Emtk-the emotion mining toolkit," in *2019 IEEE/ACM 4th International Workshop on Emotion Awareness in Software Engineering (SEmotion)*. IEEE, 2019.

[28] B. Felbo, A. Mislove, A. Søgaard, I. Rahwan, and S. Lehmann, "Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 1615–1625.

[29] H. Batra, N. S. Punn, S. K. Sonbhadra, and S. Agarwal, "Bert-based sentiment analysis: A software engineering perspective," in *International Conference on DEXA*, 2021.

[30] E. Biswas, M. E. Karabulut, L. Pollock, and K. Vijay-Shanker, "Achieving reliable sentiment analysis in the software engineering domain using bert," in *2020 IEEE ICSME*, 2020.

[31] R. Kamath, A. Ghoshal, S. Eswaran, and P. B. Honnavalli, "Emoroberta: An enhanced emotion detection model using roberta," in *IEEE International Conference on Electronics, Computing and Communication Technologies*, 2022.

[32] C. Liu, M. Osama, and A. De Andrade, "Dens: A dataset for multi-class emotion analysis," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 6293–6298.

[33] T. Wolf, L. Debut, V. Sanh, J. Chaumond, Delangue *et al.*, "Huggingface's transformers: State-of-the-art natural language processing," *arXiv preprint arXiv:1910.03771*, 2019.

[34] J. D. M.-W. C. Kenton and L. K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of NAACL-HLT*, 2019, pp. 4171–4186.

[35] 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," *arXiv preprint arXiv:1907.11692*, 2019.

[36] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray *et al.*, "Training language models to follow instructions with human feedback," *Advances in Neural Information Processing Systems*, vol. 35, pp. 27 730–27 744, 2022.

[37] R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang, and T. B. Hashimoto, "Stanford alpaca: An instruction-following llama model," 2023.

[38] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[39] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, E. Li, X. Wang, M. Dehghani, S. Brahma *et al.*, "Scaling instruction-finetuned language models," *arXiv preprint arXiv:2210.11416*, 2022.

[40] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information processing & management*, vol. 45, no. 4, pp. 427–437, 2009.

[41] J. Kocoń, I. Cichecki, O. Kaszyca, M. Kochanek, D. Szydło, J. Baran, J. Bielaniewicz, M. Gruza, A. Janz, K. Kanclerz *et al.*, "Chatgpt: Jack of all trades, master of none," *Information Fusion*, p. 101861, 2023.

[42] Y. Bang, S. Cahyawijaya, N. Lee, W. Dai, D. Su, B. Wilie, H. Lovenia, Z. Ji, T. Yu, W. Chung *et al.*, "A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity," *arXiv preprint arXiv:2302.04023*, 2023.

[43] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. Bang, A. Madotto, and P. Fung, "Survey of hallucination in natural language generation," *ACM Computing Surveys*, 2022.

[44] S. Poria, N. Majumder, D. Hazarika, D. Ghosal, R. Bhardwaj, S. Y. B. Jian, P. Hong, R. Ghosh, A. Roy, N. Chhaya *et al.*, "Recognizing emotion cause in conversations," *Cognitive Computation*, vol. 13, pp. 1317–1332, 2021.

[45] E. Turcan, S. Wang, R. Anubhai, K. Bhattacharjee, Y. Al-Onaizan, and S. Muresan, "Multi-task learning and adapted knowledge models for emotion-cause extraction," in *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, 2021, pp. 3975–3989.

[46] X. Li, W. Gao, S. Feng, D. Wang, and S. Joty, "Span-level emotion cause analysis by bert-based graph attention network," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 3221–3226.

[47] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 2002, pp. 311–318.

[48] Y. Chen, S. Y. M. Lee, S. Li, and C.-R. Huang, "Emotion cause detection with linguistic constructions," in *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, 2010, pp. 179–187.

[49] Y.-T. Lin, A. Papangelis, S. Kim, S. Lee, D. Hazarika, M. Namazifar, D. Jin, Y. Liu, and D. Hakkani-Tur, "Selective in-context data augmentation for intent detection using pointwise v-information," in *Proceedings of the 17th Conference of the European Chapter of the ACL*, 2023, pp. 1455–1468.

[50] M. Denkowski and A. Lavie, "Choosing the right evaluation for machine translation: an examination of annotator and automatic metric performance on human judgment tasks," in *Proceedings of the 9th Conference of the Association for Machine Translation in the Americas: Research Papers*, 2010.

[51] S. Seljan, M. Brkić, and T. Vičić, "BLEU evaluation of machine-translated English-Croatian legislation," in *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, May 2012, pp. 2143–2148.

[52] Z. Gao, X. Xia, J. Grundy, D. Lo, and Y.-F. Li, "Generating question titles for stack overflow from mined code snippets," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 29, no. 4, pp. 1–37, 2020.

[53] B. Morgan and C. Jensen, "Lessons learned from teaching open source software development," in *Open Source Software: Mobile Open Source Technologies: 10th IFIP WG 2.13 International Conference on Open Source Systems, OSS 2014, San José, Costa Rica, May 6-9, 2014. Proceedings*, vol. 10. Springer Berlin Heidelberg, 2014, pp. 305–310.

[54] M. R. Wrobel, "Emotions in the software development process," in *2013 6th International Conference on Human System Interactions (HSI)*. IEEE, 2013, pp. 518–523.

[55] K. Gelbrich, "Anger, frustration, and helplessness after service failure: coping strategies and effective informational support," *Journal of the Academy of Marketing Science*, vol. 38, pp. 567–585, 2010.

[56] M.-A. Storey and C. Treude, "Software engineering dashboards: Types, risks, and future," in *Rethinking Productivity in Software Engineering*. Springer, 2019, pp. 179–190.

[57] M. Guizani, T. Zimmermann, A. Sarma, and D. Ford, "Attracting and retaining oss contributors with a maintainer dashboard," in *Proceedings of the 2022 ACM/IEEE 44th International Conference on Software Engineering: Software Engineering in Society*, 2022, pp. 36–40.

[58] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[59] L. Villarroel, G. Bavota, B. Russo, R. Oliveto, and M. Di Penta, "Release planning of mobile apps based on user reviews," in *Proceedings of the 38th ICSE*, 2016.

[60] S. Scalabrino, G. Bavota, B. Russo, M. Di Penta, and R. Oliveto, "Listening to the crowd for the release planning of mobile apps," *IEEE Transactions on Software Engineering*, vol. 45, no. 1, pp. 68–86, 2017.

[61] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "Dbscan revisited, revisited: why and how you should (still) use dbscan," *ACM Transactions on Database Systems (TODS)*, vol. 42, no. 3, pp. 1–21, 2017.

[62] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction*. Springer, 2009, vol. 2.

[63] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," *arXiv preprint arXiv:1908.10084*, 2019.

[64] M. Maguire and B. Delahunt, "Doing a thematic analysis: A practical, step-by-step guide for learning and teaching scholars." *All Ireland Journal of Higher Education*, vol. 9, no. 3, 2017.

[65] B. Saunders, J. Sim, T. Kingstone, S. Baker, J. Waterfield, B. Bartlam, H. Burroughs, and C. Jinks, "Saturation in qualitative research: exploring its conceptualization and operationalization," *Quality & quantity*, vol. 52, 2018.

[66] G. Kudrjavets, A. Kumar, N. Nagappan, and A. Rastogi, "Mining code review data to understand waiting times between acceptance and merging: An empirical analysis," in *Proceedings of the 19th International Conference on Mining Software Repositories*, 2022, pp. 579–590.

[67] O. Parry, G. M. Kapfhammer, M. Hilton, and P. McMinn, "Surveying the developer experience of flaky tests," in *Proceedings of the 44th International Conference on Software Engineering: Software Engineering in Practice*, 2022, pp. 253–262.

[68] D. G. Widder, M. Hilton, C. Kästner, and B. Vasilescu, "A conceptual replication of continuous integration pain points in the context of travis ci," in *Proceedings of the 2019 27th acm joint meeting on european software engineering conference and symposium on the foundations of software engineering*, 2019, pp. 647–658.

[69] G. Berman, "Machine learning practices and infrastructures," in *Proceedings of the 2023 AAAI/ACM Conference on AI, Ethics, and Society*, 2023, pp. 466–481.

[70] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2021, pp. 4582–4597.

[71] A. Lazaridou, E. Gribovskaya, W. Stokowiec, and N. Grigorev, "Internet-augmented language models through few-shot prompting for open-domain question answering," *arXiv preprint arXiv:2203.05115*, 2022.

[72] K. Zhou, J. Yang, C. C. Loy, and Z. Liu, "Learning to prompt for vision-language models," *International Journal of Computer Vision*, vol. 130, 2022.

[73] 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 transactions on pattern analysis and machine intelligence*, vol. 41, no. 9, pp. 2251–2265, 2018.

[74] J. Wei, M. Bosma, V. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le, "Finetuned language models are zero-shot learners," in *ICLR*, 2022. [Online]. Available: https://openreview.net/forum?id=gEZrGCozdqR

[75] R. Zhong, K. Lee, Z. Zhang, and D. Klein, "Adapting language models for zero-shot learning by meta-tuning on dataset and prompt collections," in *Findings of the Association for Computational Linguistics: EMNLP 2021*, 2021, pp. 2856–2878.

[76] L. Zhang, T. Xiang, and S. Gong, "Learning a deep embedding model for zero-shot learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2021–2030.

[77] H. Lin, P. Yi, J. Ma, H. Jiang, Z. Luo, S. Shi, and R. Liu, "Zero-shot rumor detection with propagation structure via prompt learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 4, 2023, pp. 5213–5221.

[78] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[79] Google. (2023) Bard. [Online]. Available: https://bard.google.com

[80] S. Zhang, H. Wu, X. Xu, G. Zhu, and M.-Y. Hsieh, "Cl-ecpe: contrastive learning with adversarial samples for emotion-cause pair extraction," *Connection Science*, vol. 34, no. 1, pp. 1877–1894, 2022.

[81] B. Xu, H. Lin, Y. Lin, and K. Xu, "Two-stage supervised ranking for emotion cause extraction," *Knowledge-Based Systems*, vol. 228, p. 107225, 2021.

[82] R. Xia and Z. Ding, "Emotion-cause pair extraction: A new task to emotion analysis in texts," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 1003–1012.

[83] S. Y. M. Lee, Y. Chen, and C.-R. Huang, "A text-driven rule-based system for emotion cause detection," in *Proceedings of the NAACL HLT 2010 workshop on computational approaches to analysis and generation of emotion in text*, 2010, pp. 45–53.

[84] L. Gui, R. Xu, D. Wu, Q. Lu, and Y. Zhou, "Event-driven emotion cause extraction with corpus construction," in *Social Media Content Analysis: Natural Language Processing and Beyond*. World Scientific, 2018, pp. 145–160.

[85] L. Gui, J. Hu, Y. He, R. Xu, Q. Lu, and J. Du, "A question answering approach for emotion cause extraction," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 1593–1602.

[86] X. Zheng, Z. Liu, Z. Zhang, Z. Wang, and J. Wang, "Ueca-prompt: Universal prompt for emotion cause analysis," in *Proceedings of the 29th International Conference on Computational Linguistics*, 2022, pp. 7031–7041.

[87] Z. Wang, Q. Xie, Z. Ding, Y. Feng, and R. Xia, "Is chatgpt a good sentiment analyzer? a preliminary study," *arXiv preprint arXiv:2304.04339*, 2023.

[88] D. Girardi, F. Lanubile, N. Novielli, and A. Serebrenik, "Emotions and perceived productivity of software developers at the workplace," *IEEE Transactions on Software Engineering*, 2021.

[89] D. Graziotin, X. Wang, and P. Abrahamsson, "Understanding the affect of developers: theoretical background and guidelines for psychoempirical software engineering," in *Proceedings of the 7th International Workshop on Social Software Engineering*, 2015, pp. 25–32.

[90] P. Wurzel Gonçalves, G. Çalikli, and A. Bacchelli, "Interpersonal conflicts during code review: Developers' experience and practices," *Proceedings of the ACM on Human-Computer Interaction*, vol. 6, no. CSCW1, pp. 1–33, 2022.

[91] A. Sajadi, K. Damevski, and P. Chatterjee, "Interpersonal trust in oss: Exploring dimensions of trust in github pull requests," in *2023 IEEE/ACM 45th International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*. IEEE, 2023, pp. 19–24.

[92] S. C. Müller and T. Fritz, "Stuck and frustrated or in flow and happy: Sensing developers' emotions and progress," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 1. IEEE, 2015, pp. 688–699.

[93] D. Graziotin, F. Fagerholm, X. Wang, and P. Abrahamsson, "On the unhappiness of software developers," in *Proceedings of the 21st international conference on evaluation and assessment in software engineering*, 2017, pp. 324–333.

[94] ——, "What happens when software developers are (un) happy," *Journal of Systems and Software*, vol. 140, pp. 32–47, 2018.

[95] C. D. Egelman, E. Murphy-Hill, E. Kammer, M. M. Hodges, C. Green, C. Jaspan, and J. Lin, "Predicting developers' negative feelings about code review," in *Proceedings of the ACM/IEEE 42nd ICSE*, 2020, pp. 174–185.

[96] I. Ferreira, J. Cheng, and B. Adams, "The" shut the f** k up" phenomenon: Characterizing incivility in open source code review discussions," *Proceedings*

of the ACM on Human-Computer Interaction, vol. 5, no. CSCW2, pp. 1–35, 2021.

[97] C. Miller, S. Cohen, D. Klug, B. Vasilescu, and C. KaUstner, "" did you miss my comment or what?" understanding toxicity in open source discussions," in *Proceedings of the 44th International Conference on Software Engineering*, 2022, pp. 710–722.

[98] R. Ehsani, R. Rezapour, and P. Chatterjee, "Exploring Moral Principles Exhibited in Software-related Text: A Case Study on GitHub Locked Issues," in *Proceedings of the 31st ACM Joint ESEC/FSE: Ideas, Visions and Reflections Track*, 2023.

[99] J. Sarker, A. K. Turzo, M. Dong, and A. Bosu, "Automated identification of toxic code reviews using toxicr," *ACM Trans. Softw. Eng. Methodol.*, feb 2023.

[100] T. Kinsman, M. Wessel, M. A. Gerosa, and C. Treude, "How do software developers use github actions to automate their workflows?" in *2021 IEEE/ACM 18th International Conference on MSR*. IEEE, 2021, pp. 420–431.

[101] R. Li, P. Pandurangan, H. Frluckaj, and L. Dabbish, "Code of conduct conversations in open source software projects on github," *Proceedings of the ACM on Human-computer Interaction*, vol. 5, no. CSCW1, pp. 1–31, 2021.

[102] G. Gousios, M.-A. Storey, and A. Bacchelli, "Work practices and challenges in pull-based development: the contributor's perspective," in *Proceedings of the 38th International Conference on Software Engineering*, 2016, pp. 285–296.