e-Informatica Software Engineering Journal
Volume 19, Issue 1, pages: 1–37
DOI: 10.37190/e-Inf250104

BibTeX

Emotion Classification on Software Engineering Q&A Websites

Didi Awovi Ahavi-Tete*, Sangeeta Sangeeta*
*Corresponding authors: didi.ahavitete@gmail.com, s.sangeeta@keele.ac.uk

Article info

Dataset link: https://drive.google.com/drive/folders/1qXyLx9OhpHVcXLMTsTYdjhxhV-t6G54j

Keywords:
empirical and experimental studies in software engineering
data mining in software engineering
prediction models in software engineering
AI and knowledge based software engineering

Submitted: 20 Nov. 2023
Revised: 2 Oct. 2024
Accepted: 6 Oct. 2024
Available online: 19 Nov. 2024

Abstract

Background. With the rapid proliferation of question-and-answer websites for software developers like Stack Overflow, there is an increasing need to discern developers' emotions from their posts to assess the influence of these emotions on their productivity such as efficiency in bug fixing.
Aim. We aimed to develop a reliable emotion classification tool capable of accurately categorizing emotions in Software Engineering (SE) websites using data augmentation techniques to address the data scarcity problem because previous research has shown that tools trained on other domains can perform poorly when applied to SE domain directly.
Method. We utilized four machine learning techniques, namely BERT, CodeBERT, RFC (Random Forest Classifier), and LSTM. Taking an innovative approach to dataset augmentation, we employed word substitution, back translation, and easy data augmentation methods. Using these we developed sixteen unique emotion classification models: *EmoClassBERT-Original*, *EmoClassRFC-Original*, *EmoClassLSTMOriginal*, *EmoClassCodeBERT-Original*, *EmoClassLSTM-Substitution*, *EmoClassBERT-Substitution*, *EmoClassRFC-Substitution*, *EmoClassCodeBERT-Substitution*, *EmoClassBERT-Translation*, *EmoClassLSTM-Translation*, *EmoClassRFC Translation*, *EmoClassCodeBERT-Translation*, *EmoClassBERT-EDA*, *EmoClassLSTM-EDA*, *EmoClassCodeBERT-EDA*, and *EmoClassRFC-EDA*. We compared the performance of this model on a gold standard state-of-the-art database and techniques (Multi-label SO BERT and EmoTxt).
Results. An initial investigation of models trained on the augmented datasets demonstrated superior performance to those trained on the original dataset. EmoClassLSTM-Substitution, EmoClassBERT-Substitution, EmoClassCodeBERT-Substitution, and EmoClassRFC-Substitution models show improvements of 13%, 5%, 5%, and 10% as compared to EmoClassLSTM-Original, EmoClassBERT-Original, EmoClassCodeBERT-Original, and EmoClassRFC-Original, respectively, in average F_1-score. The *EmoClassCodeBERT-Substitution* performed the best and outperformed the Multi-label SO BERT and Emotxt by 2.37% and 21.17%, respectively, in average F_1-score. A detailed investigation of the models on 100 runs of the dataset shows that BERT-based and CodeBERT-based models gave the best performance. This detailed investigation reveals no significant differences in the performance of models trained on augmented datasets and the original dataset on multiple runs of the dataset.
Conclusion. This research not only underlines the strengths and weaknesses of each architecture but also highlights the pivotal role of data augmentation in refining model performance, especially in the software engineering domain.

1. Introduction

Software engineering (SE) is a domain that, while inherently technical, is deeply influenced by human factors such as emotions, cognitive biases, and decision-making processes. These human-centric aspects play a crucial role in shaping the dynamics of software development, from team collaborations to the end product's quality [1]. The emotional undertones evident in various communication channels, whether in code comments, pull requests, or interactive developer forums, can provide insights into the effect of developers [2]. They can highlight potential misunderstandings, pinpoint areas that spark contention, or even forecast the emergence of software bugs and vulnerabilities [3, 4]. Such insights, if harnessed correctly, can be instrumental in anticipating issues and enhancing overall software development efficiency.

Previous research shows that emotion can greatly impact various software development activities. For example, positive emotion can improve job satisfaction and productivity [5]. Experiments by Girardi et al. [6] show that positive emotions occur when developers work on implementing new features. However, their results also show that negative emotions are triggered in developers when they encounter unexpected code behavior and missing documentation. It can also be caused by time pressure or being stuck with the task. A study by Graziotin et al. [7] shows possible consequences of positive and negative emotions. For example, their study shows that positive emotion leads to several positive consequences like high code quality, high motivation, higher creativity, etc., whereas negative emotion causes various negative consequences like low productivity, low participation, and work withdrawal. A study by Novielli et al. [8], shows that negative emotion can also lead to difficulty in learning new programming languages. All of the above examples indicate the importance of correctly recognizing the emotions of software developers.

The above research shows that emotion recognition is important for various software development tasks. However, it is found to be very challenging because of data scarcity issues. In the software engineering domain, there is limited availability of the ground truth or manually annotated data because manual annotation is resource resource-intensive task [9] [10]. Also, there are researches that show that emotion classification models trained on a dataset of other domains do not perform well when used in the software engineering domain [11]. Advancements in natural language processing (NLP) have unveiled powerful models like BERT, CodeBERT, LSTM networks, and ensemble methods like RFC. These models have demonstrated state-of-the-art results in various NLP tasks [12, 13], prompting exploration into their potential for emotion classification within the SE realm [14, 15]. Yet, one perennial challenge in machine learning (ML) and NLP tasks is the need for extensive and diverse training datasets [2]. Hence, there is a need to address this data scarcity issue. In this paper, we focus on improving the performance of emotion classification in the SE domain using the data augmentation technique.

Data augmentation, a technique of artificially enhancing the dataset size and variability, has shown promising results in improving model robustness and generalization [16]. Among various data augmentation techniques, word substitution and back translation have garnered attention for their ability to retain semantic integrity while introducing syntactic variability [2, 16, 17]. Additionally, Kufakou et al. [18] show that the easy data augmentation approach gave the best results in their experiment. This study aims to investigate the efficacy of data augmentation techniques with machine learning algorithms in the context of emotion classification in the SE domain. In this research, we utilized four machine learning techniques, namely Bidirectional Encoder Representations from Transformers (BERT), CodeBERT, Long

Short-Term Memory (LSTM) neural network, and the Random Forest Classifier (RFC) model. We used three data argumentation techniques: *word substitution*, *back translation*, and *Easy Data Augmentation (EDA)*. Using these we developed sixteen unique emotion classification models: *EmoClassBERTOriginal*, *EmoClassCodeBERT-Original*, *EmoClassRFC-Original*, *EmoClassLSTM-Original*, *EmoClassLSTM-Substitution*, *EmoClassBERT-Substitution*, *Emo ClassCodeBERT-Substitution*, *EmoClassRFC-Substitution*, *EmoClassBERT-Translation*, *EmoClassCodeBERT-Translation*, *EmoClassLSTM-Translation*, *EmoClassRFC-Translation*, *EmoClassBERT-EDA*, *EmoClassCodeBERT-EDA*, *EmoClassLSTM-EDA*, and *EmoClass RFC-EDA*. We evaluated the performance of the proposed model(s) on a gold-standard state-of-the-art database [19]. We compared its performance with state-of-art techniques Multi-label SO BERT and EmoTxt [14]. Specifically, we answer the following research questions in this study:

- **RQ1: Which classification model performs better between LSTM, BERT, CodeBERT, and RFC?** Experimental results show that the BERT and codeBERT model outperformed LSTM and RFC in emotion classification.
- **RQ2: An initial investigation: Can data augmentation improve the model's performance?** Experimental results show that models trained on the augmented datasets demonstrated superior performance to those trained on the original dataset. EmoClassLSTM-Substitution, EmoClassBERT-Substitution, EmoClassCodeBERT-Substitution, and EmoClassRFC-Substitution models show improvements of 13%, 5%, and 10% as compared to EmoClassLSTM-Original, EmoClassBERT-Original, and EmoClassRFC-Original, respectively, in average F_1-score.
- **RQ3: How do EmoClassLSTM, EmoClassBERT, EmoClassCodeBERT, and EmoClassRFC compare to existing tools?** The *EmoClassCodeBERT-Substitution* performed best and outperformed the Multi-label SO BERT and Emotxt by 2.37% and 21.17%, respectively, in average F_1-score.
- **RQ4: How does algorithm randomness affect the performance of the proposed models?** The BERT-based and CodeBERT models perform best for emotion classification. There is no significant difference in the performance of models trained on augmented and non-augmented data.

By bridging the advanced NLP techniques with the unique challenges and intricacies of SE texts, this research hopes to contribute a robust methodology for emotion recognition in this vital domain.

2. Background

In today's interconnected world, a vast number of individuals across the globe are utilizing various online platforms like blogs, forums, and social media sites to express their thoughts and share opinions. In the SE domain, online communities and channels have become prominent platforms for individuals to express their views and share experiences. SE communities, which include forums, chat groups, and dedicated platforms like GitHub[1] and Stack Overflow[2], serve as virtual gathering spaces for developers, programmers, and information technology project managers. These channels have emerged as valuable hubs of knowledge, where professionals discuss coding practices and issues [20]. Consequently,

[1]https://github.com/
[2]https://stackoverflow.com/

a substantial amount of valuable data is generated within these communities, forming a rich source of insights into the thoughts, opinions, and challenges software engineers worldwide face.

Liu [21] describes sentiment analysis, also known as opinion mining, as a discipline intersecting natural language processing, text mining, and computational linguistics. It evaluates the emotional tone of texts from diverse sources like social media, e-commerce sites, and blogs. This analysis aids organizations in discerning public sentiment, understanding product perceptions, and detecting emerging trends [22]. Emotion classification, also known as affective computing, is a subfield of sentiment analysis that focuses on identifying, understanding, and interpreting human emotions [23]. While sentiment analysis classifies the feelings expressed in a text into three categories: positive, negative, and neutral, emotion classification goes further to recognize a wide range of human emotions, including *Joy*, *Anger*, *Sadness*, *Surprise*, *Disgust*, and *Fear* [24].

The SE field is not only technical but also deeply human, involving collaboration, creativity, and problem-solving [25]. Emotions, like *Sadness*, *Anger*, and *Joy*, play a pivotal role in influencing productivity, team dynamics, and decision-making in SE [1, 26]. SE researchers have been employing sentiment analysis techniques as discussed in several applications [9, 15, 27]. For example, Murgia et al. [28] observed that issue reports carry emotions. Ortu et al. [3] reported that emotions could influence team communication, decision-making, and problem-solving strategies, thereby significantly affecting the software development process. They showed there is a correlation between emotion expressed in issue comments and bug-fixing productivity. They mined emotions from 560 000 Jira comments, revealing that expressions of *Joy* and *Love* correlated with faster issue resolutions, while *Sadness* was linked to longer delays. Understanding and addressing these emotions is essential for fostering a positive and productive work environment. Uddin et al. [29] mined the Application Programming Interface (API) discussion from StackOverflow and reported that sentiments can be used to predict pros and cons related to the adoption of APIs. Several studies use sentiment to detect issues in applications' reviews [30]. Gu et al. [31] analyzed sentiment in user reviews. They proposed the SUR-Miner model which helps classify user reviews into one of the predefined classes like aspect evaluation, bug reports, feature requests, praise, and others. SUR-Miner's ability to discern and categorize user feedback into predefined classes significantly enhances the analysis and interpretation of user sentiments when evaluating applications. Panichella et al. [32], used sentiment analysis techniques combined with natural language processing and text analysis to classify user reviews into the following classes: Information Giving, Information Seeking, Feature Request, and Problem Discovery. Their approach proves valuable for pinpointing problem areas in the software and directing efforts towards resolving those identified bugs. Furthermore, this method helps to quickly spot areas requiring improvement, empowering developers to swiftly address these issues and deploy new functionalities that align with end users' preferences and needs. Rahman et al. [33] used opinion mining to recommend insightful comments from source code on StackOverflow.

Despite the progress made in the field, Imran et al. [2], reported the unsuitability of state-of-the-art emotion categorization tools on SE data. The research also highlighted how the tool's accuracy decreases when trained on one communication channel and assessed on another. Hence, from this, we can say that emotion classification on SE Q&A websites is still a developing field of study. This research aims to develop an emotion classification algorithm capable of effectively identifying the emotions of software developers on SE communication channels to investigate and implement techniques for improving the accuracy

and generalization of the prediction model. We use the gold standard (manually) annotated dataset[3] extracted from Stack Overflow extracted by Novielli et al. [19] for this research. By leveraging NLP, the preprocessing tasks were performed followed by the implementation of data augmentation techniques. We developed three emotion classification algorithms using the LSTM, the BERT, and the RFC model. The models were evaluated against the Multi-label Stack Overflow BERT model and EmoTxt presented in [14].

3. Related work

The recognition of the role of emotion in SE has gained substantial momentum within the academic and industrial communities in recent years. This section aims to provide an in-depth review of the literature relating to this topic, surveying the progression and future trajectories of this field.

3.1. Sentiment analysis in software engineering

In their research, Jongeling et al. [34] conducted an in-depth evaluation of the performance of two widely used sentiment analysis tools, namely SentiStrength [35] and NLTK [36]. Their analysis initially included four tools but ultimately focused on SentiStrength and NLTK. They used seven datasets for their investigation, including issue trackers and questions from Stack Overflow, a popular online platform for the programming community. Their findings highlighted a significant challenge when applying these sentiment analysis tools to SE contexts. Both SentiStrength and NLTK were initially developed for non-SE domains, which have substantial differences in language and sentiment expression compared to texts in the SE field. Their observations underscore the need for sentiment analysis tools specifically designed and trained for the unique characteristics of SE texts.

Guzman et al. [27] adopted a lexical-based technique to analyze the sentiments expressed in 60425 commit comments of 29 OSS projects. SentiStrength was used to convert emotions expressed in commit comments into quantitative values. SentiStrength allocates specific scores to tokens listed in a dictionary, which also encompasses common emoticons. Words expressing negative sentiments are assigned a value ranging between $[-5, -1]$, while those expressing positive sentiments receive a value between $[1, 5]$. Words with neutral sentiment are assigned values of 1 and -1. On the other hand, extreme sentiment expressions, words with very positive and negative feelings, are given scores of 5 and -5, respectively. A commit comment is considered to be positive if its overall emotion score falls within the range of $[1, 5]$, negative if the score is in the $[-1, -5]$ range, and neutral if the score lies within the $[-1, 1]$ range. Furthermore, an analysis was conducted on the correlation between these quantified emotions and various factors such as the programming language used, the team distribution, and others. The researchers emphasized looking beyond the average emotion score of the committed messages. They recommended considering both average positive and negative scores, and the spread of positive, negative, and neutral documents for a deeper understanding of the emotional content.

To overcome the limitations associated with SentiStrength, Islam and Zibran [37] implemented SentiStrength-SE, a sentiment analysis tool built upon SentiStrength (lexical-based

[3]https://github.com/collab-uniba/EmotionDatasetMSR18/blob/master/Emotions_GoldSandard_and Annotation.xlsx

approach) and specifically tailored to the SE domain. This tool integrates an understanding of the nuances and jargon used in the field, enabling it to interpret sentiment more accurately than general sentiment analysis tools. SentiStrength-SE proved superior to the original SentiStrength tool when evaluated using a substantial dataset. This dataset has 5600 issue comments from various SE projects.

The research led by Ahmed et al. [15] resulted in the creation of SentiCR, a specialized sentiment analysis tool for SE. This development came about due to the inadequacy of the existing tools they evaluated using their dataset of 2000 code review comments from 20 Open Source Software projects. SentiCR was developed using the Python programming language, incorporating the Natural Language Toolkit (NLTK) for language preprocessing tasks. Then, the scikit-learn library was employed for the supervised learning algorithms. As part of the data preprocessing tasks, the Term Frequency – Inverse Document Frequency (TF-IDF) method was used for feature extraction, and then eight supervised learning algorithms were evaluated. These include Adaptive Boosting, Decision Tree, Gradient Boosting Tree, Naive Bayes, Random Forest, Multilayer Perceptron, Support Vector Machine with Stochastic Gradient Descent and Linear Support Vector Machine. The researchers observed that the Gradient Boosting Tree performed better than other models, with an accuracy of 82%.

Calefato et al. [38] introduced Senti4SD, a sentiment polarity classifier. Over 4000 manually annotated posts from Stack Overflow served as the training and testing basis for the classifier. Senti4SD's semantic features are derived from a distributional semantic model (DSM) that utilizes word embedding. The DSM was established by executing Word2vec on a corpus of more than 20 million documents sourced from Stack Overflow, thereby generating word vectors that encapsulate the communication style of developers. Senti4SD, trained using Support Vector Machines (SVM), overcame the problem of negative bias prevalent in existing sentiment analysis tools by combining lexicon-based, keyword-based, and semantic features. Negative bias refers to the phenomenon where texts that are actually neutral in tone are incorrectly identified as expressing negative emotions. Notably, a 19% improvement in precision for the negative class and a 25% improvement in recall for the neutral class were observed when compared with SentiStrength.

In contrast to the aforementioned studies, our research focuses on developing an emotion recognition model tailored to the software engineering domain, with the unique ability to classify and differentiate between specific emotions. By doing so, we aim to provide a nuanced and domain-specific understanding of emotional states within the context of software development, which can have significant implications for improving the overall SE processes and work environment.

3.2. Emotion classification in software engineering

Identifying specific emotions, rather than just general sentiment, offers a richer understanding of software engineers' emotional states. This detailed perspective aids in grasping team dynamics, decision-making, and productivity [39]. For example, spotting frustration may indicate task challenges, while joy or satisfaction could signify successful teamwork or development. Responding to this need, Calefato et al. [40] proposed EmoTxt, an open-source toolkit tailored for emotion detection in text. It was trained on two key datasets: 4800 Stack Overflow posts created for the study, and 4000 Jira comments from Ortu et al. [3]. EmoTxt uses six binary classifiers to detect specific emotions: *Joy*, *Love*, *Sadness*, *Anger*, *Surprise*, and *Fear*. Utilizing a supervised learning approach with Support Vector Machines (SVM), it effectively identifies emotional patterns in written content.

Murgia et al. [20] developed classifiers for each emotion category (*Love*, *Joy*, *Sadness*, and *Neutral*), with each classifier calculating the probability of a particular emotion being present in a comment. They created five versions of each classifier using SVM, Naive Bayes (NB), Single Layer Perceptron (SLP), K-Nearest Neighbor (KNN), and Random Forest (RF). Using bootstrap validation on Apache Software Foundation project comments, the SVM classifiers proved most effective for detecting love, joy, and sadness, warranting further examination. The SVM models' performance was later assessed on a separate test set of comments.

Recognizing the limitations of traditional ML techniques, researchers began to explore more advanced methods, particularly Deep Learning (DL) algorithms. DL techniques are especially suitable for complex tasks such as emotion classification because they can handle high-dimensional data and capture intricate patterns within the data. Bleyl and Buxton [14] implemented BERT models for emotion recognition in Stack Overflow comments drawing on Novielli, et al's. [19] dataset. Due to the dataset's imbalance, they augmented underrepresented emotion classes. They also fine-tuned BERT for the SE context by adding 993 prevalent technical words and emoticons from Stack Overflow to BERT's tokenizer vocabulary. Then, leveraging Masked Language Modelling, they trained BERT on a large dataset of unlabeled Stack Overflow comments and fine-tuned it on the Stack Overflow annotated dataset. Their multi-label BERT model outperformed other models.

Our research builds upon previous studies by investigating various machine learning architectures and techniques to improve the overall performance of emotion recognition models in the context of software engineering. In this paper, we explored data augmentation methods for enhancing the robustness and generalization capabilities of our emotion recognition models. In this paper, we explored data augmentation methods for enhancing the robustness and generalization capabilities of our emotion recognition models. Imran et al. [2], proposed data augmentation-based techniques for emotion classification on the SE dataset to address the data scarcity issue. They report an improvement of 9.3% in micro F_1-score as compared to popular SE tools. However, they explore only 3 types of data augmentation techniques: Unconstrained, lexicon, and polarity-based. They used a stacked approach for data augmentation. They used a stacked approach for data augmentation. In this, we focus on using simple data augmentation techniques like "back translation" to find out their effectiveness for emotion classification in the software engineering domain. In addition, Imran et al. [2], use existing emotion classification models like ESEM-E, EMTk, and SEntiEmoji whereas, in the paper, we checked the efficiency of 3 classifiers LSTM, Radom Forest, and BERT for emotion classification on the augmented dataset. The approach used in this paper extends the work done by Imran et al. [2] by adding one more dimension of using data augmentation for emotion. classification

4. Methodology

Figure 1 provides an overview of the methodology for this study. We used the Stack Overflow dataset provided by Novielli et al. [19]. This is the gold standard dataset. We use Python programming language for implementing various machine learning libraries. We notice that this dataset is highly imbalanced in nature. Hence, we explored the uses of data augmentation methods for improving the accuracy of emotion detection. We developed three main emotion classification algorithms using RF, LSTM and BERT models. We

then compare the performance of these different approaches. Finally, we evaluate how the implemented emotion classification algorithms compare to existing tools in the SE domain.

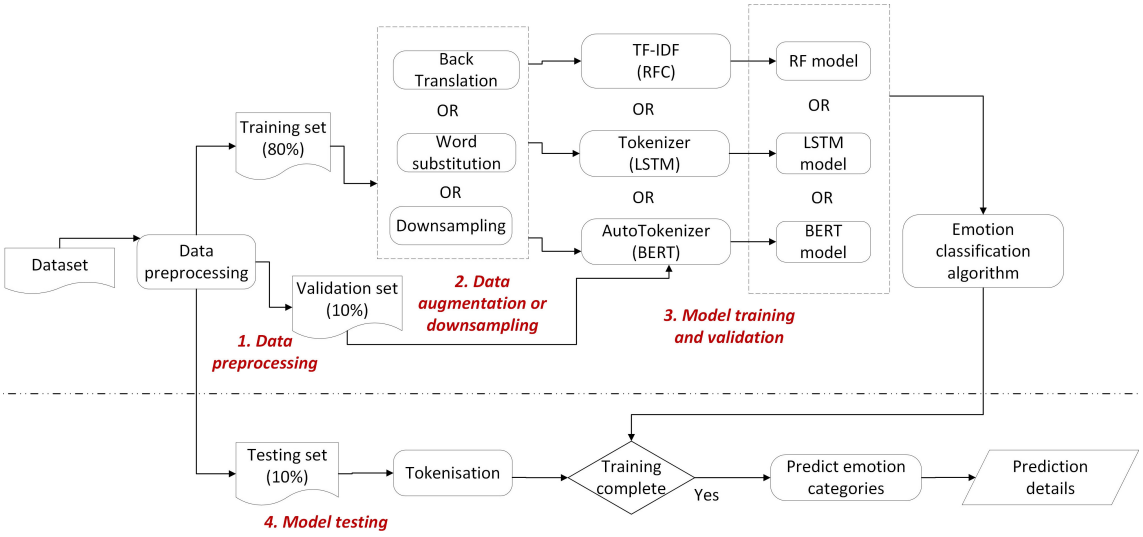

Figure 1. Overview of the methodology

4.1. Dataset description

The dataset contains 4800 Stack Overflow entries, encompassing questions, answers, and comments, and is a sample from the unlabeled Stack Overflow dataset of June 2008 to September 2015 [19]. Part of the Stack Exchange network of Q&A websites, Stack Overflow is a popular Q&A site for software developers. On Stack Overflow, users can ask questions, answer questions, vote on questions and answers, and earn reputation and badges. As discussed by Novielli et al. [19], the dataset was annotated by a group of twelve volunteers. Each entry received annotations from three different individuals, focusing on the six fundamental emotions (*Love*, *Joy*, *Surprise*, *Anger*, *Sadness*, and *Fear*). Determining the emotion for an observation relied on a majority consensus approach. If at least two of the three evaluators identified a specific emotion for an observation, then that emotion was assigned to the sample. Table 1 shows, an example of the dataset. However, not every observation-emotion combination was labeled, and some observations were labeled with more than one emotion. Approximately 56% of the comments are labeled with just one emotion, 6% are marked with two or more emotions, and the remaining comments are without emotion labels [14]. For this study, any post not annotated with emotion was regarded as devoid of emotion and, therefore, classified as neutral posts. This dataset is organized into individual worksheets for each emotion label: *Love*, *Joy*, *Surprise*, *Anger*, *Sadness*, and *Fear* [19]. The worksheets were, therefore, merged into a single sheet and saved as a CSV file.

Table 1. Examples from Novielli et al. dataset [19]

Text	Rater 1	Rater 2	Rater 3	Gold label
SVG transform on text attribute works excellent! This snippet, for example, will increase your text by 2× at Y-axis.	X		X	LOVE
Excellent! This is exactly what I needed. Thanks!	X	X	X	LOVE
Have added a modern solution as of May 2014 in answers below.				
Have you tried removing "preload" attribute? (Afraid I can't be much help otherwise!)				

Table 2. Emotion label distribution

Number of observations conveying the emotion							Total
Love	Joy	Surprise	Anger	Sadness	Fear	Neutral	
1181	488	43	867	227	103	1918	4735

4.2. Data preprocessing

4.2.1. Text cleaning techniques

After merging the worksheets, some duplicates were identified in the dataset. Duplicate entries can lead to biased or skewed results because they do not represent unique instances of the data. The duplicated observations were removed from the experimental dataset. We removed duplicates from the dataset using a two-step process: 1) automated text matching and 2) manual verification. We removed a total of 65 duplicate entries (Love: 39, Joy: 3, Surprise: 2, Anger: 15, Sadness: 3, Fear: 3). We also notice the presence of some irrelevant attributes in the dataset such as information about the group, set, id, and raters. We removed all these attributes from the experimental dataset. The label Neutral was assigned to the entries not annotated in the original dataset. Table 2 shows the number of instances for each emotion category.

Removal of non-alphabetic characters. Non-alphabetic characters were removed as part of an essential approach designed to streamline raw textual data. This curtails the presence of excessive symbols, punctuations, and numbers seen as noise, which can add meaningless variability. By filtering out such characters, the resultant text not only becomes more readable but also more concise. This makes it more compatible with the strict requirements of computational processing and linguistic analysis, leading to a more efficient and accurate prediction algorithm [41].

Case folding. Furthermore, the entire text corpus was converted to lowercase to ensure the homogeneity of the dataset. This is because the words "Analysis" and "analysis", though semantically identical, would be processed as separate tokens due to their case difference. Such distinctions introduce redundancies, thereby increasing the dimensionality of the data without adding meaningful variance [42]. Using consistent casing in the dataset ensures that the text is standardized. This standardization is vital for constructing consistent and reproducible models that can generalize effectively to unseen data, thereby enhancing the reliability of the prediction model.

Stop words, stemming and lemmatization. While it is usually essential to remove stop words when handling NLP tasks, in line with previous studies [38], stop words were not removed, as comments such as "I am happy with this output" and "I am not happy with this output" express different emotions. Neither Stemming nor lemmatization was performed since, according to Calefato et al., [38], a varied form might convey useful information.

4.2.2. Tokenization

Tokenization, in NLP, is breaking down the text into smaller pieces, known as "tokens". While these tokens are commonly individual words, they can also be sentences, parts of words, or even single characters [43]. The type of token selected is usually based on the particular NLP task. As an example, tokenizing the phrase "I love coding" results in ["I", "love", "coding"]. Tokenization is crucial because, before text data can be analyzed or fed into machine learning algorithms, it often must be transformed from its raw form into a structured format [44]. The Keras, BERT, and scikit-learn libraries were utilized to tokenize the Stack Overflow posts. The tokenization process was handled differently for the three models in the NLP task. For the LSTM model, the tokenizer module from the Keras library was used to process the posts, while the BERT tokenizer was used for the BERT model. The Term Frequency-Inverse Document Frequency (TF-IDF) was used to extract features for the RFC model. This essential step was carried out to convert the text into numerical form and to aid in building the vocabulary for the dataset.

4.3. Text exploratory analysis

Text Exploratory Analysis (TEA) aims to meticulously decipher the embedded structure, recurrent patterns, and potential aberrations within the dataset [45].

4.3.1. Sentiment polarity

Sentiment polarity in text analysis evaluates the overall sentiment or tone of a piece of writing. It categorizes the sentiment as positive, negative, or neutral, allowing for a quick assessment of the general mood of the expressed thoughts [46]. For instance, a statement such as *"Excellent, I'm glad that worked for you!"* would likely be categorized as possessing a positive polarity, whereas *"This is one of the shortcomings of DGV that I absolutely hate and why I almost always bind to an IEnumerable of an anonymous type."* would be attributed a negative polarity. A statement such as, *"I understand that server-side validation is an absolute must to prevent malicious users (or simply users who choose to disable javascript) from bypassing client-side validation"* might be considered neutral. We used TextBlob to obtain a brief overview of the sentiment polarity within the dataset. TextBlob, a Python library rooted in NLTK and Pattern, offers lexicon-based sentiment analysis by producing polarity and subjectivity scores. Polarity scores range from -1 (negative) to 1 (positive), with 0 being neutral. Subjectivity scores span from 0 (factual) to 1 (opinion-based).

Figure 2 showcases the sentiment polarity distribution in the dataset, detailing percentages of positive, negative, and neutral sentiments for an overall mood assessment. However, sentiment polarity provides a generalized perspective, missing the detailed layers of emotion

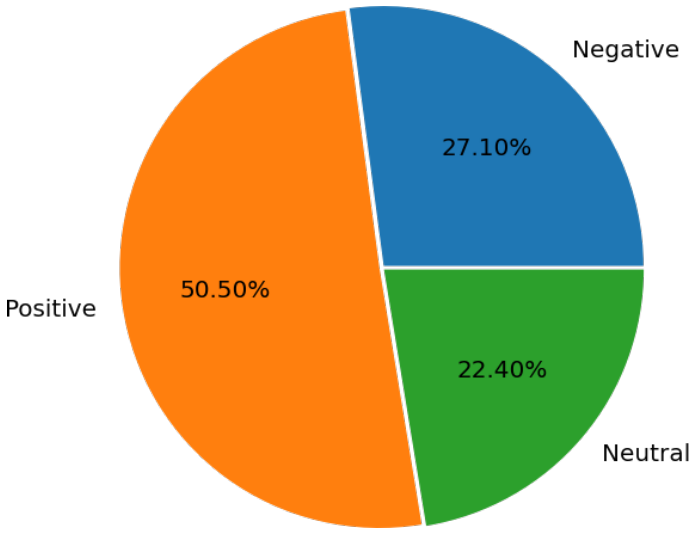

Figure 2. Sentiment polarity in the dataset

detection. Unlike broad labels of positive, negative, or neutral, emotion detection identifies specific feelings like *Joy*, *Anger*, *Love*, *Fear*, *Sadness* and *Surprise*.

4.3.2. Distribution of emotion categories

This was performed to understand the distribution, quality, and structure of the emotion-labeled dataset. Understanding the distribution of emotions in the dataset is crucial since class imbalance can introduce biases into the ML models [47]. By visualizing the distribution, one can take necessary measures to augment the data for under-represented categories or use techniques to address imbalances during model training. The graph presented in Figure 3 provides a visualization of the distribution of various emotion categories in the dataset. The bar chart shows the count of instances for each emotion category, while the pie

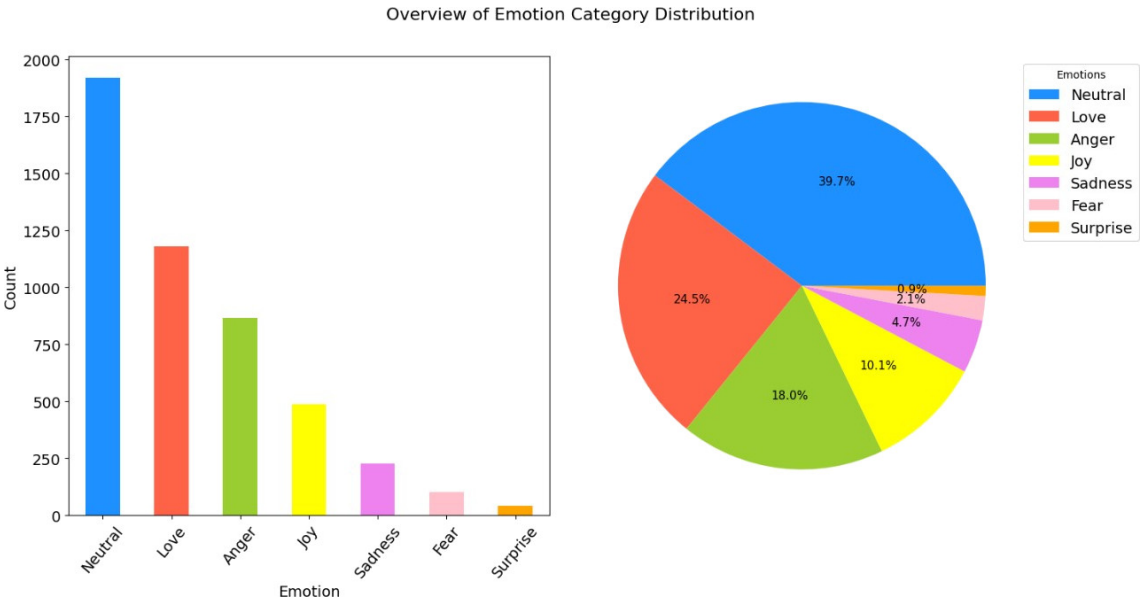

Figure 3. Overview of emotion category distribution

chart illustrates the proportion of each class. A quick analysis of this visualization reveals a pronounced imbalance among the different types of emotions. The dataset does not adequately represent certain emotion categories, namely *Joy*, *Sadness*, *Fear*, and *Surprise*.

4.4. Addressing the data imbalance

In ML, data imbalance pertains to the uneven distribution of classes within a dataset. It is a prevalent challenge, especially in classification tasks, where certain classes are significantly underrepresented compared to others. This skewed representation often leads to suboptimal model performance, as the algorithms tend to exhibit a bias towards the majority class, consequently neglecting the minority class [47]. To address this imbalance in our dataset, we considered various strategies and opted for under-sampling the majority class and apply text augmentation for the minority classes.

4.4.1. Under-sampling the majority class

Under-sampling involves decreasing the number of observations from the predominant class to achieve a more balanced class representation [48]. Specifically, we randomly selected 950 samples from the Neutral emotion category. We selected this number through experimentation. We notice that the algorithm is giving better results when the data points in the neutral category have a similar number of points as in the other categories. After the neutral category, the second highest number of data points were present in the "love" category, i.e., 945. Hence, we selected 950 samples for "neutral" category.

4.4.2. Text augmentation using a contextual word embedding with BERT

Another approach adopted in this study to balance the dataset is data augmentation. This involves creating new data by slightly altering existing samples, thereby artificially enlarging the dataset. Especially, text augmentation with word substitution was performed on the minority classes. Word substitution, an effective technique to augment textual data, refers to the process of replacing words in a text with other words while aiming to retain the overall meaning or intent of the original text [20]. Before the text augmentation, the dataset was split into training, validation, and testing sets in a stratified ratio of 80–10–10 using the scikit-learn library. Only the training dataset was enhanced through augmentation, ensuring that the validation and testing sets reflect real-world situations.

Leveraging the ContextualWordEmbsAug class from the nlpaug library, 94% of the Surprise, 44% of the Sadness and 91% of the Fear emotion category samples were randomly chosen and augmented using the bert-base-uncased model. To have a relatively balanced dataset and to avoid introducing noise in the dataset, each sample from the Surprise, Sadness, and Fear categories were augmented four, one and two times, respectively. This was performed to introduce variety in the increased dataset while still preserving the original meaning.

For each sample, randomly selected words were replaced by the nearest words derived from the embedding space provided by the BERT model. Unlike traditional word embeddings, which give every word a fixed vector representation regardless of its context in a sentence, contextual embeddings adjust word representations based on the surrounding words in a given sentence, making it suitable for text augmentation [49]. Tables 3, 4, and 5 show the distribution of the categories in the augmented training dataset, test dataset, and

Table 3. Distribution of the emotion categories in the training set, test set, and validation set for word substitution

	Emotion category						
	Love	Anger	Joy	Sadness	Fear	Surprise	Neutral
Number of samples (Training set)	945	694	390	262	232	162	760
Number of samples (Test set)	118	87	49	23	10	4	95
Number of samples (Validation set)	118	86	49	22	11	5	95

Table 4. Distribution of the emotion categories in the training set, test set, and validation set using back translation

	Emotion category						
	Love	Anger	Joy	Sadness	Fear	Surprise	Neutral
Number of samples (Training set)	945	694	390	262	157	66	760
Number of samples (Test set)	118	87	49	23	10	4	95
Number of samples (Validation set)	118	86	49	22	11	5	95

Table 5. Distribution of the emotion categories in the training set, test set, and validation set using easy data augmentation approach

	Emotion category						
	Love	Anger	Joy	Sadness	Fear	Surprise	Neutral
Number of samples (Training set)	945	694	390	262	232	162	760
Number of samples (Test set)	118	87	49	23	10	4	95
Number of samples (Validation set)	118	86	49	22	11	5	95

validation dataset using word substitution, back translation, and easy data augmentation technique, respectively. For the Word substitution and EDA methods, the selected samples in the Surprise, Fear, and Sadness categories were augmented 4, 2, and 1 times, respectively. However, the actual number of samples in the EDA-augmented training set may be lower for certain folders. This discrepancy occurred because the code used was unable to process some rows. For the Back translation approach, the selected samples were augmented only once to prevent duplicates in the augmented training set.

4.5. Emotion classification algorithms

We employed RF, LSTM, and BERT to develop the emotion classification algorithms. The section below presents the description and architecture of the three models.

4.5.1. LSTM

We chose LSTM for this emotion classification task because it is particularly adept at processing sequence data and learning long-term dependencies, which is often inherent in language-based tasks. Moreover, LSTM excels at detecting complex patterns within natural language (NL), patterns that other models might not [50], and has proved reliable for NL understanding tasks like text classification [51, 52] and sentiment analysis [53]. LSTM, a type of Recurrent Neural Network (RNN), was introduced [54] to overcome the vanishing and exploding gradients problem that RNNs suffer from. LSTM networks have

four main components: the input gate, forget gate, output gate, and memory cell. The memory cell holds relevant information, with the gates managing data intake, retention, and output [54]. This architecture enables LSTMs to manage long sequences efficiently, making them particularly effective for text classification and capturing intricate human emotions in text [55].

4.5.2. BERT transformer model

Researchers have started exploring the use of transformer models for sentiment analysis and emotion classification tasks [14, 56]. Thanks to their architecture and pre-training on extensive corpora, they are adept at detecting subtle nuances in textual data. Developed by researchers at Google in 2018 [57], BERT represents a significant advancement in the NLP domain known for its bidirectional understanding of language and has paved the way for models RoBERTa, FlauBERT. The transformer architecture, presented by Vaswani et al. [58], utilizes self-attention mechanisms for contextual understanding, processing words concurrently for efficiency. BERT, building on this, discerns context by masking and predicting certain input tokens, thereby enhancing linguistic representations. BERT employs token, segment, and positional embeddings for input representation. It uses WordPiece tokenization to manage out-of-vocabulary words and maintains input data sequence by integrating these embeddings [57]. Initially, BERT was offered in two versions:

– BERT-BASE with 12 layers, 768 hidden sizes, 12 attention heads, and 110 million parameters.
– BERT-LARGE with 24 layers, 1024 hidden sizes, 16 attention heads, and 340 million parameters.

BERT's pre-training involved the Masked Language Model (MLM) and Next Sentence Prediction (NSP) tasks. In MLM, BERT predicts concealed input tokens, while in NSP, it identifies sentence sequences, aiding question-answering tasks [57].

4.5.3. Random Forest Classifier

Many researchers have used RFC for text classification tasks [59]. Specifically, studies like the one by [15] have shown that Random Forest is one of the reliable models for detecting sentiment in posts on Q&A websites for software developers. Furthermore, we selected RFC because of its capability to train on small datasets, as is the case in this study. Introduced adequately by Breiman in [60], RFC is an ensemble learning method that creates numerous decision trees during its training phase and merges their results for more accurate and reliable predictions [60]. Each tree makes its own classification decision based on the input data. The final class determination for a given input is achieved by taking a majority vote from the classifications of all individual trees.

5. Evaluation metrics

After developing a machine learning model, it is essential to use evaluation metrics to determine its performance on previously unseen test data. We selected popular metrics used for the classification task.

5.1. Precision, recall, and *F*-score

While the balanced accuracy score can provide an overview of the model performance, it does not show the performance of each class in the dataset. Precision provides insight into a model's correct predictions for each class. For class i, precision is:

$$Precision_i = \frac{TP_i}{TP_i + FP_i}$$

where TP_i is the number of correctly predicted instances of class i, and FP_i is instances wrongly predicted as class i. Recall evaluates the model's ability to identify all possible positive instances within the dataset [61]. It is determined by the following formula, where FN_i denotes instances of class i wrongly predicted as another class.

$$Recall_i = \frac{TP_i}{TP_i + FN_i}$$

F_1-score provides a harmonic mean of the two metrics, and is computed as follows:

$$F_1\text{-}score_i = \frac{2 * Precision_i * Recall_i}{Precision_i + Recall_i}$$

6. Results

This section provides an overview of the results from various experiments conducted on the Stack Overflow dataset. The investigation comprises three primary research queries addressed in the paragraphs below. We made all the dataset and source code publicly available for replication: https://drive.google.com/drive/folders/1qXyLx9OhpHVcXLMTsTYdjhxhV-t6G54j.

6.1. RQ1: Which classification model performs best among LSTM, BERT, CodeBERT, and RFC?

Motivation. With the rapid advancements in ML and NLP, many models have been proposed to solve classification problems in text data. Among these, the RFC, Support Vector Machines (SVM) have been commonly employed. These algorithms have demonstrated their capacity to yield reliable classification results across diverse contexts. In recent years, researchers have used more sophisticated tools, such as LSTM [52], BERT [56], CodeBERT [62], and RFC [59]. In this RQ, we compare the performance of LSTM, BERT, CodeBERT, and RFC in classifying emotions within the Stack Overflow dataset.

Approach. In this part, we give a detailed description of the parameters used for all the algorithms.

LSTM. After the preprocessing techniques, the LSTM model was implemented using the `keras` and `tensorflow` libraries. Textual data was tokenized and normalized to sequences of integers with a uniform length of 195 and the labels were one-hot encoded. The LSTM model includes an embedding layer converting input to a 128-dimensional vector and a two-layered LSTM: the first layer with 128 neurons (and 0.2 dropout rate) and the second with 64 neurons. The dropout parameter helps to prevent overfitting, whereas the 2-layered

LSTM structure allows the model to capture more complex patterns. The model's output layer has 7 units with softmax activation for multi-class classification. It is compiled using the `categorical_crossentropy` loss function with the `adam` optimizer. The model is set to train for a maximum of 20 epochs using a batch size of 64 and the `class_weight` parameter which handles the class imbalance. However, the early stopping criteria could terminate it prematurely. Regularization techniques like `EarlyStopping` and `ModelCheckpoint` were used to monitor the validation loss and ensure the model stops training if there is no improvement after 7 epochs. Several optimization strategies were employed to enhance the model's performance and prevent overfitting. We explored a variety of hyperparameters (dropout rate, number of neurons in the layers, number of epochs) to fine-tune the model. Different hyperparameter combinations were tested to determine which gave the highest results.

BERT. The emotion classification model, built with BERT, utilized the same augmented training, validation, and testing datasets as the LSTM neural network. It was developed using the `bert-base-uncased` pre-trained model and the `datasets`, `scikit-learn`, `torch`, and `transformers` libraries; and trained on a Graphics Processing Unit (GPU). The model was trained for 4 epochs – to avoid overfitting while still allowing it to detect the emotion contained in the comment – and enhanced with the Adam with Weight Decay (`adamw_torch`) optimizer, which is a renowned gradient descent optimization algorithm for transformers models. Before the tokenization, the data frames are converted to HuggingFace's Dataset format, and subsequently mapped to a DatasetDict (Dataset dictionary) object. Each text entry is tokenized, padded, and truncated using the AutoTokenizer function of the bert-base-uncased model. Tokenization is crucial as it converts the input data into a format the model can understand. Meanwhile, padding and truncating ensure that all input sequences have the same length, a requirement for batch processing in neural networks. Training parameters such as the number of epochs (4), learning rate (2×10^{-5}), batch size (16), optimizer (`adamw_torch`), and others are set using the `TrainingArguments` class. These parameters play a pivotal role in guiding the model's learning behavior. The number of epochs dictates how many times the model reviews the entire dataset, the learning rate determines the step size when updating weights, and the batch size indicates the number of data points processed simultaneously. The `Trainer` class from the Transformers library is used to train the model on the training dataset while validating it on the validation dataset. The HuggingFace Trainer class simplifies the process of training machine learning models by encapsulating the necessary training tasks, making it both efficient and user-friendly.

CodeBERT. It is a bimodal pre-trained model developed using transformer-based neural architecture. It is designed for NL-PL applications such as natural language code search and documentation generation. The model is trained using the hybrid objective function. This uses both bimodal and uni-modal data for model training. The bimodal data provides the input token and the uni-modal data is used for learning better generators. As recommended by the authors in [62][4], we used the `RobertaTokenizer` for tokenizing our input data. Then, we fine-tuned the `microsoft/codebert-base` model on our dataset utilizing the same training parameters as those used for the BERT model we previously developed. This approach was taken to enable a direct performance comparison between the two transformer-based models. By keeping the parameters consistent, we ensured that any differences in performance could be attributed to the models themselves, rather than variations in the training process.

[4]https://github.com/microsoft/CodeBERT

Table 6. Performance of the models

	EmoClassLSTM-Original			EmoClassBERT-Original			EmoClassCodeBERT-Original			EmoClassRFC-Original		
Emotion	Precision	Recall	F_1-score	Precision	Recall	F_1-score	Precision	Recall	F_1-score	Precision	Recall	F_1-score
Love	0.72	0.70	0.71	0.76	0.84	**0.80**	0.81	0.79	**0.80**	0.72	0.78	0.75
Joy	0.35	0.39	0.37	0.50	0.45	0.47	0.54	0.65	**0.59**	0.53	0.35	0.42
Surprise	0.07	0.50	**0.12**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Anger	0.63	0.22	0.32	0.75	0.75	**0.75**	0.75	0.68	0.71	0.72	0.72	0.72
Sadness	0.38	0.13	0.19	0.68	0.65	**0.67**	0.59	0.70	0.64	0.68	0.57	0.62
Fear	0.18	0.20	0.19	0.58	0.70	0.64	0.70	0.70	**0.70**	0.00	0.00	0.00
Neutral	0.40	0.58	0.47	0.79	0.75	**0.77**	0.76	0.77	0.76	0.59	0.73	0.65

RFC. The `scikit-learn` library was used to implement the RFC model using the same training, validation, and testing sets as the previous models. The training set was utilized for training the model, while the validation set was employed for fine-tuning the hyperparameters and determining the best combination for the model. Finally, the testing set was used to evaluate the model's performance. For feature extraction, the code uses the TF-IDF method to convert the text data into numerical features. While the Term Frequency computes the frequency of words in a document, the Inverse Document Frequency calculates the importance of a word. Together, the TF-IDF method captures words' significance in the dataset while diminishing the weight of frequently occurring but potentially uninformative words [63]. Subsequently, the RFC is instantiated and trained on the TF-IDF processed training data using 300 trees (`n_estimators=300`). The decision to utilize 300 trees was made to strike a harmonious balance. With too few trees, the model might not capture all the nuances in the data. Conversely, an excessive number could lead to computational inefficiencies without significantly improving performance. Furthermore, we used the `GridSearchCV` technique to obtain the optimal set of parameters for the vectorizer and RFC model. Finally, the performance of the model is evaluated on the test set.

Using the above parameters we created the following models:

- **EmoClassLSTM-Original.** an LSTM model trained with the parameters described above.
- **EmoClassBERT-Original.** a BERT model trained with the parameters described above.
- **EmoClassCodeBERT-Original.** a CodeBERT model trained with the parameters described above.
- **EmoClassRFC-Original.** an RFC model trained with the parameters described above.

Results. To gain a more comprehensive insight into the model's performance, precision, recall, and the F_1-score were computed for each emotion category. These metrics offer a nuanced understanding of how well the model identifies and classifies each emotion. Table 6 details the results of each model in identifying a specific emotion category. For example, for emotion category *Joy*, EmoClassBERT-Original achieved an F_1-score of 59%, while EmoClassBERT-Original, EmoClassLSTM-Original, and EmoClassRFC-Original gave an F_1-score of 47%, 37%, and 42%, respectively. Based on the obtained results, BERT outperformed the other models in detecting most emotion categories, while EmoClassLSTM-Original and EmoClassCodeERT-Original achieved better performance for *Surprise* and *Fear*, respectively. BERT's superior performance can be attributed to its bidirectional architecture, which enables it to grasp both past and future context, and its extensive

pre-training on vast text corpora allows it to understand language nuances deeply. On the other hand, while the LSTM produced significant results compared to other emotion detection tools in the SE domain, as detailed in [2], its unidirectional processing of sequences and the limited dataset, could be factors contributing to its average inferior performance compared to BERT and RFC.

Given these results, BERT emerges as a more optimal choice for tasks that require a profound understanding of context, especially in complex datasets like Stack Overflow comments. Nevertheless, the low score for the Surprise from all the models could be due to insufficient samples and the complexity in detecting the Surprise emotion, as noted by [14].

6.2. RQ2: An initial investigation: Can data augmentation improve the model's performance?

Motivation. Training machine learning models to decipher the complex world of human emotions requires vast amounts of data, particularly labelled data that indicates which emotion is present in a given comment. In the context of Stack Overflow, this becomes even more intricate given the specific lexicon used. Given these challenges, procuring an adequately representative dataset for emotion classification on Stack Overflow is not just resource intensive but also requires extensive domain-specific knowledge to annotate the data accurately. This complexity, combined with the need for large-scale data to train robust models, leads to an intriguing proposition: Could data augmentation be utilized to synthetically expand and diversify the dataset, instead of solely relying on manual data collection and annotation?

The choice to explore Random Forest Classifier (RFC) and Long Short-Term Memory (LSTM) models alongside BERT and CodeBERT was driven by a desire to evaluate various approaches and assess their performance comprehensively. While BERT indeed demonstrated superior performance, considering alternative models allowed us to provide a more nuanced understanding of the dataset and its characteristics. The motivation for incorporating RFC and LSTM models aimed to explore how these models would handle the enriched dataset. This approach provides a broader perspective on the robustness and adaptability of different models to variations in data volume and complexity. Hence, we tested 16 combinations of various classifiers for exhaustive testing.

Table 7. Examples of data augmented using back translation from the dataset

S No.	Approach	Comment
1.	Original	million unique visitors per hour? Wow! Is this Experts Exchange or some pr n site
	Translation	millions **of** unique visitors per hour wow is this exchange **of experts** or **another site** pr n
2	Original	wow would have expected a quick answer on this well found my own answer ….
	Translation	wow would have **waited for** a quick answer on this **property** found my own answer…

Approach. Data augmentation techniques were employed to increase the diversity of the dataset. Data augmentation is an established strategy in machine learning, which

Table 8. Examples of data augmented using word substituted from the dataset

S No.	Approach	Comment
1.	Original	a unit test should do the same thing every time that it runs otherwise you may run into a situation where the unit test only fails occasionally…
	Substitution	a unit test should do the same **then** every time that it runs otherwise you may run into another situation where the unit test **still** fails occasionally…
2.	Original	I m very sorry about my horrible English only for this example I use radio button…
	Substitution	**we** m very sorry concerning my **short** english only **at** this example I use **one** button ..

can significantly enhance the quality and versatility of datasets without the need for additional data collection [2]. The dataset was divided into training, validation, and testing sets using a ratio of 80–10–10. To ensure that the validation and testing sets mirrored real-world applications, only the training set was augmented. The text augmentation methods evaluated and implemented for the underrepresented categories include:

- **Back Translation.** The BackTranslationAug class was employed for the translation. This technique involves translating a text into a secondary language (in our case, French was chosen due to its rich linguistic structure) and then reverting it to its original language, English. This often results in texts that maintain their core sentiment but are structurally or lexically varied. Table 7 shows some examples of the original and augmented datasets.
- **Word substitution.** To introduce lexical diversity, words were replaced with their closest synonyms in the contextual embedding space. While the overall sentiment remains intact, this technique ensures that the model is not biased towards specific wordings. The ContextualWordEmbsAug class was employed with the substitute action parameter for this augmentation process. Table 8 shows some examples of the original and augmented datasets.
- **Easy data augmentation.** Easy data augmentation for a given sentence performs one of the four operations randomly, i.e., synonym replacement, random insertion, swap, and random deletion.

Leveraging the nlpaug library, 94% of the Surprise, 44% of the Sadness and 91% of the Fear emotion category samples were randomly chosen and augmented. For the word substitution augmentation technique, each sample from the Surprise, Sadness and Fear categories was augmented four, one, and two times, respectively. This augmentation was done to prevent the introduction of duplicate entries in the training data and to maintain a balanced distribution across the various emotion categories. Through trial runs, we observed that excessively augmenting the dataset did not contribute to increased diversity. For the back translation approach, samples were translated only once to prevent introducing duplicates in the augmented data. These strategies were adopted to introduce variety in the augmented data while still preserving the original meaning.

For every augmentation method employed, the enhanced dataset was merged with the original training data. Once combined, this consolidated data was then provided to the machine learning model for training. Three different versions of LSTM, BERT, CodeBERT, and RFC models were developed utilizing the original and augmented training sets. To ensure consistency and optimal learning, each LSTM model was trained for a duration of 20 epochs using a batch size of 64 with regularization techniques. On the other hand, each BERT model was trained for 4 epochs with a batch size of 16 and a learning rate set

Table 9. Details of the emotion classification models implemented

Model	Augmentation technique	Reference
LSTM	Word Substitution	EmoClassLSTM-Substitution (EmoClassLSTM-S)
	Back Translation	EmoClassLSTM-Translation (EmoClassLSTM-T)
	Easy data augmentation	EmoClassLSTM-EDA (EmoClassLSTM-E)
	None	EmoClassLSTM-Original (EmoClassLSTM-O)
BERT	Word Substitution	EmoClassBERT-Substitution (EmoClassBERT-S)
	Back Translation	EmoClassBERT-Translation (EmoClassBERT-T)
	Easy data augmentation	EmoClassBERT-EDA (EmoClassBERT-E)
	None	EmoClassBERT-Original (EmoClassBERT-O)
CodeBERT	Word Substitution	EmoClassCodeBERT-Substitution (EmoClassCodeBERT-S)
	Back Translation	EmoClassCodeBERT-Translation (EmoClassCodeBERT-T)
	Easy data augmentation	EmoClassCodeBERT-EDA (EmoClassCodeBERT-E)
	None	EmoClassCodeBERT-Original (EmoClassCodeBERT-O)
RFC	Word Substitution	EmoClassRFC-Substitution (EmoClassRFC-S)
	Back Translation	EmoClassRFC-Translation(EmoClassRFC-T)
	Easy data augmentation	EmoClassRFC-EDA (EmoClassRFC-E)
	None	EmoClassRFC-Original (EmoClassRFC-O)

at 2×10^{-5}, striking a balance between speed and prediction performance, and each RFC model utilized the same hyperparameters as described earlier. Table 9 provides a summary of the models implemented.

Results. The performance of emotion detection tools for each specific emotion is outlined in Table 10 and Table 11. Table 10 shows the performance of all the models with and without augmentation whereas Table 11 shows the average F_1-score for all the models. Specifically, for the BERT models, there is not much difference between precision and recall, suggesting that these models are equally adept at identifying true positive cases (precision) as they are at capturing the total positive instances (recall). This balance is crucial in emotion detection, as it means that the model is accurate in its predictions and minimizes the risk of missing out on instances where a specific emotion is present. For the CodeBERT model, the models trained on the augmented dataset outperformed the models trained on the original dataset in most cases. However, the results for the LSTM model emphasize the importance of having sufficient training data or training it on a more balanced dataset. In most cases, the RFC models performed better than the LSTM models, indicating that RFC is more suitable for smaller datasets. The highest F_1-scores, highlighted in bold, show that:

- EmoClassBERT-Substitution performed best for *Love*;
- EmoClassCodeBERT-Original performed best for *Joy*;
- EmoClassBERT-Translation and EmoClassCodeBERT-Translation performed best for *Anger*;
- EmoClassBERT-Original performed best for *Sadness*;
- EmoClassBERT-Translation, EmoClassRFC-Substitution, and EmoClassRFC-EDA performed best for *Surprise*;
- EmoClassCodeBERT-Translation give the best F_1-score for (*Neutral*).

From these findings, several implications can be derived. The efficacy of word substitution in introducing variance through word substitution makes it suitable for a wide range of emotions. Table 11 shows that substitution-based models outperformed the other models. EmoClassBERT-substitution gives the highest F_1-score of 63%. The substitution

method shows a considerable improvement as compared to the original models, for example, EmoClassLSTM-Substitution, EmoClassBERT-Substitution, EmoClassCodeBERT-Substitution, and EmoClassRFC-Substitution models show improvements of 13%, 5%, 5%, and 10% as compared to EmoClassLSTM-Original, EmoClassBERT-Original, and EmoClassRFC-Original, respectively, in average F_1-score. On the other hand, the nuanced linguistic changes introduced by back translation make it effective for emotions like Surprise and Anger, as was demonstrated. The reduced F_1-score for the *Surprise* category might be due to its smaller sample size. Additionally, the nature of Surprise as an emotion is inherently ambiguous [9], often intertwining with both positive and negative emotions, further complicating its classification [14]. Nevertheless, EmoClassRFC-Substitution, EmoClassRFC-EDA, and EmoClassBERT-Translation outperformed other models in detecting the *Surprise* category with an F_1-score of 0.33. While data augmentation techniques showed effectiveness in various emotional categories, they did not consistently outperform the original model. Interestingly, the model performed best at detecting the emotion *Joy* when trained on non-augmented data. However, models trained on the original dataset on average underperformed compared to models trained on the augmented dataset, except the CodeBERT models (refer to Table 11). This underscores that augmentation's effectiveness can vary depending on the particular emotion under study.

Table 10. Performance of all variants of LSTM, BERT, CodeBERT and RFC

Emotion	Base model	Model	Precision	Recall	F_1-score
Love	LSTM	EmoClassLSTM-S	0.79	0.55	0.65
		EmoClassLSTM-T	0.84	0.39	0.53
		EmoClassLSTM-E	0.75	0.70	0.73
		EmoClassLSTM-O	0.72	0.70	0.71
	BERT	EmoClassBERT-S	0.81	0.81	0.81
		EmoClassBERT-T	0.82	0.77	0.79
		EmoClassBERT-E	0.81	0.85	**0.83**
		EmoClassBERT-O	0.76	0.84	0.80
	CodeBERT	EmoClassCodeBERT-S	0.79	0.85	0.82
		EmoClassCodeBERT-T	0.79	0.78	0.79
		EmoClassCodeBERT-E	0.82	0.79	0.80
		EmoClassCodeBERT-O	0.81	0.79	0.80
	RFC	EmoClassRFC-S	0.75	0.80	0.77
		EmoClassRFC-T	0.72	0.79	0.75
		EmoClassRFC-E	0.74	0.80	0.77
		EmoClassRFC-O	0.72	0.78	0.75
Joy	LSTM	EmoClassLSTM-S	0.30	0.55	0.39
		EmoClassLSTM-T	0.24	0.39	0.31
		EmoClassLSTM-E	0.42	0.41	0.41
		EmoClassLSTM-O	0.35	0.39	0.37
	BERT	EmoClassBERT-S	0.50	0.49	0.49
		EmoClassBERT-T	0.43	0.53	0.47
		EmoClassBERT-E	0.52	0.49	0.51
		EmoClassBERT-O	0.50	0.45	0.47
	CodeBERT	EmoClassCodeBERT-S	0.57	0.49	0.53
		EmoClassCodeBERT-T	0.49	0.55	0.52
		EmoClassCodeBERT-E	0.49	0.59	0.54
		EmoClassCodeBERT-O	0.54	0.65	**0.59**

Table 10 continued

Emotion	Base model	Model	Precision	Recall	F_1-score
Joy	RFC	EmoClassRFC-S	0.53	0.33	0.41
		EmoClassRFC-T	0.55	0.37	0.44
		EmoClassRFC-E	0.51	0.39	0.44
		EmoClassRFC-O	0.53	0.35	0.42
Surprise	LSTM	EmoClassLSTM-S	0.20	0.25	0.22
		EmoClassLSTM-T	0.20	0.43	0.22
		EmoClassLSTM-E	0.00	0.00	0.00
		EmoClassLSTM-O	0.07	0.50	0.12
	BERT	EmoClassBERT-S	0.33	0.25	0.29
		EmoClassBERT-T	0.50	0.25	**0.33**
		EmoClasBERT-E	0.00	0.00	0.00
		EmoClassBERT-O	0.00	0.00	0.00
	CodeBERT	EmoClassCodeBERT-S	0.33	0.25	0.29
		EmoClassCodeBERT-T	0.00	0.00	0.00
		EmoClassCodeBERT-E	0.00	0.00	0.00
		EmoClassCodeBERT-O	0.00	0.00	0.00
	RFC	EmoClassRFC-S	0.50	0.25	**0.33**
		EmoClassRFC-T	0.00	0.00	0.00
		EmoClassRFC-E	0.50	0.25	**0.33**
		EmoClassRFC-O	0.00	0.00	0.00
Anger	LSTM	EmoClassLSTM-S	0.57	0.63	0.60
		EmoClassLSTM-T	0.69	0.51	0.58
		EmoClassLSTM-E	0.66	0.54	0.59
		EmoClassLSTM-O	0.63	0.22	0.32
	BERT	EmoClassBERT-S	0.73	0.76	0.75
		EmoClassBERT-T	0.74	0.78	**0.76**
		EmoClassBERT-E	0.73	0.74	0.73
		EmoClassBERT-O	0.75	0.75	0.75
	CodeBERT	EmoClassCodeBERT-S	0.71	0.74	0.72
		EmoClassCodeBERT-T	0.76	0.76	**0.76**
		EmoClassCodeBERT-E	0.74	0.75	0.74
		EmoClassCodeBERT-O	0.75	0.68	0.71
	RFC	EmoClassRFC-S	0.72	0.71	0.72
		EmoClassRFC-T	0.76	0.74	0.75
		EmoClassRFC-E	0.73	0.72	0.73
		EmoClassRFC-O	0.72	0.72	0.72
Sadness	LSTM	EmoClassLSTM-S	0.52	0.52	0.52
		EmoClassLSTM-T	0.27	0.43	0.33
		EmoClassLSTM-E	0.57	0.0.57	0.57
		EmoClassLSTM-O	0.38	0.13	0.19
	BERT	EmoClassBERT-S	0.72	0.57	0.63
		EmoClassBERT-T	0.67	0.52	0.59
		EmoClassBERT-E	0.64	0.61	0.62
		EmoClassBERT-O	0.68	0.65	0.67
	CodeBERT	EmoClassCodeBERT-S	0.68	0.65	0.67
		EmoClassCodeBERT-T	0.67	0.61	0.64
		EmoClassCodeBERT-E	0.67	0.61	0.64
		EmoClassCodeBERT-O	0.59	0.70	0.64

Table 10 continued

Emotion	Base model	Model	Precision	Recall	F_1-score
Sadness	RFC	EmoClassRFC-S	0.67	0.52	0.59
		EmoClassRFC-T	0.65	0.57	0.60
		EmoClassRFC-E	0.71	0.65	**0.68**
		EmoClassRFC-O	0.68	0.57	0.62
Fear	LSTM	EmoClassLSTM-S	0.50	0.40	0.44
		EmoClassLSTM-T	0.40	0.20	0.27
		EmoClassLSTM-E	0.17	0.30	0.21
		EmoClassLSTM-O	0.18	0.20	0.19
	BERT	EmoClassBERT-S	0.70	0.70	0.70
		EmoClassBERT-T	0.54	0.70	0.61
		EmoClassBERT-E	0.67	0.80	0.73
		EmoClassBERT-O	0.58	0.70	0.64
	CodeBERT	EmoClassCodeBERT-S	0.73	0.80	**0.76**
		EmoClassCodeBERT-T	0.53	0.80	0.64
		EmoClassCodeBERT-E	0.64	0.70	0.67
		EmoClassCodeBERT-O	0.70	0.70	0.70
	RFC	EmoClassRFC-S	0.33	0.40	0.36
		EmoClassRFC-T	0.10	0.10	0.10
		EmoClassRFC-E	0.00	0.00	0.00
		EmoClassRFC-O	0.00	0.00	0.00
Neutral	LSTM	EmoClassLSTM-S	0.67	0.57	0.61
		EmoClassLSTM-T	0.50	0.69	0.58
		EmoClassLSTM-E	0.58	0.67	0.62
		EmoClassLSTM-O	0.40	0.58	0.47
	BERT	EmoClassBERT-S	0.72	0.75	0.74
		EmoClassBERT-T	0.80	0.75	0.77
		EmoClassBERT-E	0.74	0.72	0.73
		EmoClassBERT-O	0.79	0.75	0.77
	CodeBERT	EmoClassCodeBERT-S	0.78	0.75	0.76
		EmoClassCodeBERT-T	0.80	0.76	**0.78**
		EmoClassCodeBERT-E	0.78	0.73	0.75
		EmoClassCodeBERT-O	0.76	0.77	0.76
	RFC	EmoClassRFC-S	0.62	0.74	0.68
		EmoClassRFC-T	0.60	0.68	0.64
		EmoClassRFC-E	0.62	0.73	0.67
		EmoClassRFC-O	0.59	0.73	0.65

In summary, a nuanced understanding of the role of data augmentation in emotion classification was provided by this study. Significant enhancements in model performance can be achieved through tailored data augmentation. However, the importance of a judicious evaluation based on the emotion in focus and the augmentation technique being employed was emphasized. On average, the substitution method gives the highest F_1-score. While word substitution performed best on average, none of the augmentation methods was identified as a one-size-fits-all solution.

Table 11. Average performance of EmoClass classifiers

Model name	Avg. F_1-score	Avg improvement (in %) as compared to the original models
EmoClassLSTM-Original	0.36	–
EmoClassLSTM-Translation	0.39	3%
EmoClassLSTM-Substitution	**0.49**	13 %
EmoClassLSTM-EDA	0.44	8%
EmoClassBERT-Original	0.58	–
EmoClassBERT-Translation	0.62	4%
EmoClassBERT-Substitution	**0.63**	5%
EmoClassBERT-EDA	0.59	1%
EmoClassCodeBERT-Original	0.6	–
EmoClassCodeBERT-Translation	0.59	−1%
EmoClassCodeBERT-Substitution	**0.65**	**5%**
EmoClassCodeBERT-EDA	0.59	−1%
EmoClassRFC-Original	0.45	–
EmoClassRFC-Translation	0.47	2%
EmoClassRFC-Substitution	**0.55**	10%
EmoClassRFC-EDA	0.51	6%

6.3. RQ3: How do EmoClassLSTM, EmoClassBERT, EmoClassCodeBERT, and EmoClassRFC compare to existing tools?

Motivation. Researchers have harnessed ML algorithms for sentiment analysis or emotion classification tasks. Techniques such as SVM (as discussed by Calefato et al. [40]) and RFC (highlighted by Murgia et al. [20] have been widely employed. The prevalent strategy involves developing One-vs-All emotion classifiers. This approach involves developing distinct binary classifiers, each dedicated to one of the six basic emotions. However, the efficacy of this approach has been challenged. Bleyl and Buxton, in their [14] study, underscored the superior effectiveness of multi-label classification tools when juxtaposed against the One-vs-All methodology. Their findings suggest that the multi-label tool provides a better performance. The current research endeavors to delve deeper into this domain by examining the proficiency of the three tools: EmoClassLSTM, EmoClassBERT, and EmoClassRFC. The objective is to critically assess their performance against existing emotion classification tools developed for the SE domain.

Approach. In our endeavor to benchmark the performance of our models against existing tools, a pivotal step was the selection of a consistent dataset for a fair evaluation. Consequently, the dataset employed in the studies of Bleyl and Buxton [14] and Calefato et al. [40] was selected. We compare our results with the Multi-label BERT model and EmoTxt detailed in [14]. **EmoTxt** is an emotion classification tool implemented in a supervised learning method using the Support Vector Machines One-vs-All approach, where a binary classifier was developed for each emotion category. **Multi-label SO BERT** is a fine-tuned version of the BERT model, created by incorporating technical texts into its tokenizer and utilizing augmented data during the model training process. We compare the state-of-the-art methods with our EmoClassLSTM-Substitution, EmoClassBERT-Substitution, EmoClassCodeBERT-Substitution, and EmoClassRFC-Substitution models (as they gave the best results, refer to RQ2 in Section 6.2 for more details).

Table 12. EmoClassLSTM, EmoClassBERT, EmoClassCodeBERT and EmoClassRFC vs. existing tools built on the same dataset

Emotion	EmoClassLSTM -Substitution F_1	EmoClassBERT -Substitution F_1	EmoClassCodeBERT -Substitution F_1	EmoClassRFC -Substitution F_1	Multi-label SO BERT F_1	EmoTxt F_1
Love	68%	81%	82%	77%	**84%**	69%
Joy	31%	49%	53%	41%	**56%**	38%
Surprise	15%	29%	29%	**33%**	26%	23%
Anger	59%	75%	72%	72%	**80%**	68%
Sadness	43%	63%	**67%**	59%	60%	52%
Fear	12%	70%	**76%**	36%	59%	6%
Average	38.00%	61.17%	**63.17%**	53.00%	60.80%	42.00%

Results. Table 12 presents the performance of EmoClassLSTM-Substitution, EmoClass-BERT-Substitution, EmoClassCodeBERT-Substitution, and EmoClassRFC-Substitution against that of the multi-label BERT model and EmoTxt reported in [14]. In Table 12, the best-performing model for each specific emotion is shown in bold. Among the models compared, Multi-label SO BERT demonstrates notable proficiency, especially in identifying the emotions of *Love*, *Joy*, and *Anger*. On the other hand, EmoClassCodeBERT-Substitution stands out when it comes to categorizing the emotions of *Sadness* and *Fear*, while EmoClassRFC-Substitution outperformed other models in detecting *Surprise* emotions in the text. On average EmoClassCdeoBERT-Substitution performed the best and gave the highest F_1-score of 63.17%. This model outperformed the Multi-label SO BERT and Emotxt by 2.37% and 21.17%, respectively.

This distinction in performance across different emotions emphasizes the importance of selecting the right model based on the specific needs of an emotion analysis task. The lower F_1-score of the *Surprise* emotion could be attributed to its low sample size and the difficulty in detecting it as reported in [14]. Overall, EmoClassCodeBERT-Substitution emerged as the top-performing emotion classification tool built with the dataset with an average of 63.17%.

6.4. RQ 4: How does algorithm randomness affect the performance of the proposed models?

Motivation. Performance evaluation of a model on only one dataset does not provide a clear indication of whether the results obtained are statistically significant or not. Hence, to address this issue, in this RQ, we performed an in-depth evaluation of the various models proposed in this paper.

Approach. We run the various data augmentation techniques for 100 iterations and generate 100 training, testing, and validation datasets [64]. We evaluated each model on these 100 datasets and reported the median values of the performance metric (i.e., F_1-score). We computed the Wilcoxon signed-rank test to compare the performances of different models. Additionally, we computed Cliff's delta [65] to quantify the difference between the two distributions.

Table 13. Median F_1-score

Emotion	Base model	Model	Median F_1-score
Love	LSTM	EmoClassLSTM-S	0.70
		EmoClassLSTM-T	0.70
		EmoClassLSTM-E	0.69
		EmoClassLSTM-O	0.69
	BERT	EmoClassBERT-S	**0.95**
		EmoClassBERT-T	**0.95**
		EmoClassBERT-E	**0.95**
		EmoClassBERT-O	**0.95**
	CodeBERT	EmoClassCodeBERT-S	0.94
		EmoClassCodeBERT-T	0.94
		EmoClassCodeBERT-E	**0.95**
		EmoClassCodeBERT-O	**0.95**
	RFC	EmoClassRFC-S	0.76
		EmoClassRFC-T	0.76
		EmoClassRFC-E	0.76
		EmoClassRFC-O	0.76
Joy	LSTM	EmoClassLSTM-S	0.35
		EmoClassLSTM-T	0.35
		EmoClassLSTM-E	0.36
		EmoClassLSTM-O	0.35
	BERT	EmoClassBERT-S	0.87
		EmoClassBERT-T	0.88
		EmoClassBERT-E	**0.89**
		EmoClassBERT-O	0.88
	CodeBERT	EmoClassCodeBERT-S	0.86
		EmoClassCodeBERT-T	0.87
		EmoClassCodeBERT-E	**0.89**
		EmoClassCodeBERT-O	0.88
	RFC	EmoClassRFC-S	0.34
		EmoClassRFC-T	0.35
		EmoClassRFC-E	0.35
		EmoClassRFC-O	0.34
Surprise	LSTM	EmoClassLSTM-S	0.13
		EmoClassLSTM-T	0.16
		EmoClassLSTM-E	0.10
		EmoClassLSTM-O	0.12
	BERT	EmoClassBERT-S	0.73
		EmoClassBERT-T	**0.75**
		EmoClassBERT-E	**0.75**
		EmoClassBERT-O	**0.75**
	CodeBERT	EmoClassCodeBERT-S	0.73
		EmoClassCodeBERT-T	0.73
		EmoClassCodeBERT-E	**0.75**
		EmoClassCodeBERT-O	**0.75**
	RFC	EmoClassRFC-S	0.00
		EmoClassRFC-T	0.00
		EmoClassRFC-E	0.00
		EmoClassRFC-O	0.00

Table 13 continued

Emotion	Base model	Model	Median F_1-score
Anger	LSTM	EmoClassLSTM-S	0.56
		EmoClassLSTM-T	0.56
		EmoClassLSTM-E	0.56
		EmoClassLSTM-O	0.57
	BERT	EmoClassBERT-S	**0.97**
		EmoClassBERT-T	**0.97**
		EmoClassBERT-E	**0.97**
		EmoClassBERT-O	**0.97**
	CodeBERT	EmoClassCodeBERT-S	**0.97**
		EmoClassCodeBERT-T	**0.97**
		EmoClassCodeBERT-E	**0.97**
		EmoClassCodeBERT-O	**0.97**
	RFC	EmoClassRFC-S	0.72
		EmoClassRFC-T	0.71
		EmoClassRFC-E	0.71
		EmoClassRFC-O	0.72
Sadness	LSTM	EmoClassLSTM-S	0.29
		EmoClassLSTM-T	0.29
		EmoClassLSTM-E	0.28
		EmoClassLSTM-O	0.30
	BERT	EmoClassBERT-S	0.88
		EmoClassBERT-T	0.89
		EmoClassBERT-E	**0.90**
		EmoClassBERT-O	0.89
	CodeBERT	EmoClassCodeBERT-S	0.87
		EmoClassCodeBERT-T	0.88
		EmoClassCodeBERT-E	0.89
		EmoClassCodeBERT-O	**0.90**
	RFC	EmoClassRFC-S	0.42
		EmoClassRFC-T	0.43
		EmoClassRFC-E	0.45
		EmoClassRFC-O	0.46
Fear	LSTM	EmoClassLSTM-S	0.14
		EmoClassLSTM-T	0.17
		EmoClassLSTM-E	0.15
		EmoClassLSTM-O	0.16
	BERT	EmoClassBERT-S	0.89
		EmoClassBERT-T	**0.90**
		EmoClassBERT-E	**0.90**
		EmoClassBERT-O	**0.90**
	CodeBERT	EmoClassCodeBERT-S	0.88
		EmoClassCodeBERT-T	**0.90**
		EmoClassCodeBERT-E	**0.90**
		EmoClassCodeBERT-O	**0.90**
	RFC	EmoClassRFC-S	0.25
		EmoClassRFC-E	0.14
		EmoClassRFC-T	0.23
		EmoClassRFC-O	0.14

Table 13 continued

Emotion	Base model	Model	Median F_1-score
Neutral	LSTM	EmoClassLSTM-S	0.53
		EmoClassLSTM-T	0.53
		EmoClassLSTM-E	0.53
		EmoClassLSTM-O	0.55
	BERT	EmoClassBERT-S	**1.00**
		EmoClassBERT-T	**1.00**
		EmoClassBERT-E	**1.00**
		EmoClassBERT-O	**1.00**
	CodeBERT	EmoClassCodeBERT-S	**1.00**
		EmoClassCodeBERT-T	**1.00**
		EmoClassCodeBERT-E	**1.00**
		EmoClassCodeBERT-O	**1.00**
	RFC	EmoClassRFC-S	0.64
		EmoClassRFC-T	0.65
		EmoClassRFC-E	0.66
		EmoClassRFC-O	0.67

Table 14. Results of Wilcoxon Rank Test and Cliff's Delta: Value: V, Practical Difference: PD, value marked as **bold*** for Wilcoxon Rank test p-value indicate that p-value > 0.5.

Comparison between		Wilcoxon result		Cliff's Delta	
		Statistic	p-value	Value	PD
EmoClassBERT-S	EmoClassBERT-T	16 051.5	1.18212×10^{-24}	−0.06769	negligible
EmoClassBERT-S	EmoClassBERT-E	12 605.0	4.27656×10^{-34}	−0.09323	negligible
EmoClassBERT-S	EmoClassBERT-O	32 752.0	1.21951×10^{-12}	−0.06890	negligible
EmoClassBERT-S	EmoClassLSTM-S	32 752.0	1.21951×10^{-12}	−0.06890	negligible
EmoClassBERT-S	EmoClassLSTM-T	0.0	4.17153×10^{-116}	0.96236	large
EmoClassBERT-S	EmoClassLSTM-E	0.0	4.17161×10^{-116}	0.96305	large
EmoClassBERT-S	EmoClassLSTM-O	3.0	2.90243×10^{-116}	0.96131	large
EmoClassBERT-S	EmoClassRFC-S	339.0	1.15648×10^{-115}	0.92741	large
EmoClassBERT-S	EmoClassRFC-T	300.0	9.29481×10^{-116}	0.92580	large
EmoClassBERT-S	EmoClassRFC-E	376.0	1.37104×10^{-115}	0.92513	large
EmoClassBERT-S	EmoClassRFC-O	349.0	1.09128×10^{-115}	0.92444	large
EmoClassBERT-S	EmoClassCodeBERT-S	21 084.5	1.87545×10^{-18}	0.07175	negligible
EmoClassBERT-S	EmoClassCodeBERT-T	28 534.5	1.38822×10^{-3}	0.01398	negligible
EmoClassBERT-S	EmoClassCodeBERT-E	14 393.0	6.22144×10^{-27}	−0.06830	negligible
EmoClassBERT-S	EmoClassCodeBERT-O	11 412.0	2.17045×10^{-36}	−0.08747	negligible
EmoClassBERT-T	EmoClassBERT-E	21 473.0	9.12036×10^{-5}	−0.02774	negligible
EmoClassBERT-T	EmoClassBERT-O	38 388.0	5.11859×10^{-1} *	−0.00416	negligible
EmoClassBERT-T	EmoClassLSTM-S	0.0	4.17150×10^{-116}	0.97394	large
EmoClassBERT-T	EmoClassLSTM-T	3.0	2.90203×10^{-116}	0.97416	large
EmoClassBERT-T	EmoClassLSTM-E	1.0	2.87717×10^{-116}	0.97453	large
EmoClassBERT-T	EmoClassLSTM-O	0.0	2.86518×10^{-116}	0.97328	large
EmoClassBERT-T	EmoClassRFC-S	6.0	2.76076×10^{-116}	0.94493	large
EmoClassBERT-T	EmoClassRFC-T	10.0	2.64816×10^{-116}	0.94356	large
EmoClassBERT-T	EmoClassRFC-E	7.0	2.79445×10^{-116}	0.94290	large
EmoClassBERT-T	EmoClassRFC-O	6.0	2.46871×10^{-116}	0.94235	large
EmoClassBERT-T	EmoClassCodeBERT-S	12 402.5	5.43028×10^{-46}	0.13961	negligible
EmoClassBERT-T	EmoClassCodeBERT-T	10 832.5	1.23414×10^{-33}	0.08178	negligible
EmoClassBERT-T	EmoClassCodeBERT-E	27 205.5	6.64069×10^{-1} *	−0.00096	negligible
EmoClassBERT-T	EmoClassCodeBERT-O	22 043.0	3.59604×10^{-5}	−0.02094	negligible
EmoClassBERT-E	EmoClassBERT-O	24 588.0	2.74271×10^{-3}	0.02283	negligible

Table 14 continued

Comparison between		Wilcoxon result		Cliff's Delta	
		Statistic	p-value	Value	PD
EmoClassBERT-E	EmoClassLSTM-S	0.0	4.17134×10^{-116}	0.97392	large
EmoClassBERT-E	EmoClassLSTM-T	0.0	4.17135×10^{-116}	0.97416	large
EmoClassBERT-E	EmoClassLSTM-E	0.0	2.86496×10^{-116}	0.97437	large
EmoClassBERT-E	EmoClassLSTM-O	0.0	2.86523×10^{-116}	0.97334	large
EmoClassBERT-E	EmoClassRFC-S	3.0	2.71017×10^{-116}	0.94491	large
EmoClassBERT-E	EmoClassRFC-T	5.0	2.56950×10^{-116}	0.94353	large
EmoClassBERT-E	EmoClassRFC-E	4.0	2.75170×10^{-116}	0.94254	large
EmoClassBERT-E	EmoClassRFC-O	4.0	2.43425×10^{-116}	0.94192	large
EmoClassBERT-E	EmoClassCodeBERT-S	11 183.5	1.40559×10^{-50}	0.16362	small
EmoClassBERT-E	EmoClassCodeBERT-T	13 700.0	8.50572×10^{-38}	0.10781	negligible
EmoClassBERT-E	EmoClassCodeBERT-E	19 343.0	1.79090×10^{-4}	0.02713	negligible
EmoClassBERT-E	EmoClassCodeBERT-O	26 083.0	7.38967×10^{-1} *	0.00699	negligible
EmoClassBERT-O	EmoClassLSTM-S	204.0	1.00153×10^{-115}	0.96252	large
EmoClassBERT-O	EmoClassLSTM-T	19.0	3.10836×10^{-116}	0.96268	large
EmoClassBERT-O	EmoClassLSTM-E	6.0	2.93978×10^{-116}	0.96326	large
EmoClassBERT-O	EmoClassLSTM-O	363.5	1.35996×10^{-115}	0.96135	large
EmoClassBERT-O	EmoClassRFC-S	775.0	7.46804×10^{-115}	0.93024	large
EmoClassBERT-O	EmoClassRFC-T	776.0	7.16232×10^{-115}	0.92869	large
EmoClassBERT-O	EmoClassRFC-E	743.0	6.56139×10^{-115}	0.92785	large
EmoClassBERT-O	EmoClassRFC-O	774.0	6.77701×10^{-115}	0.92725	large
EmoClassBERT-O	EmoClassCodeBERT-S	28 564.5	1.34261×10^{-26}	0.13898	negligible
EmoClassBERT-O	EmoClassCodeBERT-T	30 526.5	1.31764×10^{-15}	0.08349	negligible
EmoClassBERT-O	EmoClassCodeBERT-E	36 832.5	6.83737×10^{-1} *	0.00298	negligible
EmoClassBERT-O	EmoClassCodeBERT-O	21 531.0	1.15393×10^{-3}	−0.01634	negligible
EmoClassLSTM-S	EmoClassLSTM-T	108 055.5	1.88391×10^{-1}	−0.02240	negligible
EmoClassLSTM-S	EmoClassLSTM-E	109 302.0	2.06887×10^{-1}	0.02461	negligible
EmoClassLSTM-S	EmoClassLSTM-O	106 642.0	1.11156×10^{-1}	−0.01544	negligible
EmoClassLSTM-S	EmoClassRFC-S	76 989.5	2.08736×10^{-17}	−0.21437	small
EmoClassLSTM-S	EmoClassRFC-T	83 650.5	4.43458×10^{-13}	−0.19516	small
EmoClassLSTM-S	EmoClassRFC-E	83 893.5	6.19643×10^{-13}	−0.20649	small
EmoClassLSTM-S	EmoClassRFC-O	91 179.5	7.52484×10^{-9}	−0.17538	small
EmoClassLSTM-S	EmoClassCodeBERT-S	215.0	7.20206×10^{-116}	−0.94840	large
EmoClassLSTM-S	EmoClassCodeBERT-T	119.0	4.77297×10^{-116}	−0.95625	large
EmoClassLSTM-S	EmoClassCodeBERT-E	0.0	4.17157×10^{-116}	−0.97449	large
EmoClassLSTM-S	EmoClassCodeBERT-O	0.0	4.17135×10^{-116}	−0.97679	large
EmoClassLSTM-T	EmoClassLSTM-E	99 377.0	2.07059×10^{-3}	0.04728	negligible
EmoClassLSTM-T	EmoClassLSTM-O	116 046.5	9.90057×10^{-1} *	0.00678	negligible
EmoClassLSTM-T	EmoClassRFC-S	80 052.0	3.63798×10^{-1}	−0.19956	small
EmoClassLSTM-T	EmoClassRFC-T	86 731.0	2.65058×10^{-11}	−0.18003	small
EmoClassLSTM-T	EmoClassRFC-E	86 810.5	2.93300×10^{-11}	−0.19122	small
EmoClassLSTM-T	EmoClassRFC-O	94 476.0	1.37314×10^{-7}	−0.16183	small
EmoClassLSTM-T	EmoClassCodeBERT-S	40.0	4.95359×10^{-11}	−0.94836	large
EmoClassLSTM-T	EmoClassCodeBERT-T	0.0	4.17170×10^{-116}	−0.95633	large
EmoClassLSTM-T	EmoClassCodeBERT-E	0.0	2.86477×10^{-116}	−0.97482	large
EmoClassLSTM-T	EmoClassCodeBERT-O	0.0	2.86491×10^{-116}	−0.97707	large
EmoClassLSTM-E	EmoClassLSTM-O	104 530.0	2.05498×10^{-2}	−0.04039	negligible
EmoClassLSTM-E	EmoClassRFC-S	75 175.0	2.43168×10^{-18}	−0.23016	small
EmoClassLSTM-E	EmoClassRFC-T	81 695.5	4.08018×10^{-14}	−0.20997	small
EmoClassLSTM-E	EmoClassRFC-E	81 649.5	2.61120×10^{-14}	−0.22027	small
EmoClassLSTM-E	EmoClassRFC-O	89 755.5	1.48505×10^{-9}	−0.18798	small
EmoClassLSTM-E	EmoClassCodeBERT-S	157.0	8.18665×10^{-116}	−0.94948	large
EmoClassLSTM-E	EmoClassCodeBERT-T	0.0	4.17173×10^{-116}	−0.95734	large
EmoClassLSTM-E	EmoClassCodeBERT-E	0.0	2.86496×10^{-116}	−0.97492	large
EmoClassLSTM-E	EmoClassCodeBERT-O	0.0	2.86489×10^{-116}	−0.97727	large
EmoClassLSTM-O	EmoClassRFC-S	78 856.0	2.67139×10^{-16}	−0.20616	small

Table 14 continued

Comparison between		Wilcoxon result		Cliff's Delta	
		Statistic	p-value	Value	PD
EmoClassLSTM-O	EmoClassRFC-T	85 162.0	2.39909×10^{-12}	−0.18758	small
EmoClassLSTM-O	EmoClassRFC-E	85 894.0	6.31752×10^{-12}	−0.19880	small
EmoClassLSTM-O	EmoClassRFC-O	93 549.5	5.27101×10^{-8}	−0.16845	small
EmoClassLSTM-O	EmoClassCodeBERT-S	14.0	3.04265×10^{-116}	−0.94738	large
EmoClassLSTM-O	EmoClassCodeBERT-T	5.0	2.92716×10^{-116}	−0.95528	large
EmoClassLSTM-O	EmoClassCodeBERT-E	0.0	2.86508×10^{-116}	−0.97385	large
EmoClassLSTM-O	EmoClassCodeBERT-O	0.0	2.86511×10^{-116}	−0.97637	large
EmoClassRFC-S	EmoClassRFC-T	86 325.0	9.50083×10^{-1} *	0.01028	negligible
EmoClassRFC-S	EmoClassRFC-E	88 114.5	3.04052×10^{-1}	0.00134	negligible
EmoClassRFC-S	EmoClassRFC-O	84 969.5	5.94841×10^{-1} *	0.01800	negligible
EmoClassRFC-S	EmoClassCodeBERT-S	1204.0	4.72830×10^{-114}	−0.90868	large
EmoClassRFC-S	EmoClassCodeBERT-T	379.0	1.38250×10^{-115}	−0.92185	large
EmoClassRFC-S	EmoClassCodeBERT-E	0.0	2.67542×10^{-116}	−0.94549	large
EmoClassRFC-S	EmoClassCodeBERT-O	0.0	2.67547×10^{-116}	−0.94745	large
EmoClassRFC-T	EmoClassRFC-E	80 746.0	3.49193×10^{-1}	−0.00890	negligible
EmoClassRFC-T	EmoClassRFC-O	77 657.0	8.74585×10^{-1} *	0.00859	negligible
EmoClassRFC-T	EmoClassCodeBERT-S	1099.0	2.92091×10^{-114}	−0.90710	large
EmoClassRFC-T	EmoClassCodeBERT-T	346.0	1.14477×10^{-115}	−0.92034	large
EmoClassRFC-T	EmoClassCodeBERT-E	0.0	2.51494×10^{-116}	−0.94391	large
EmoClassRFC-T	EmoClassCodeBERT-O	0.0	2.51493×10^{-116}	−0.94602	large
EmoClassRFC-E	EmoClassRFC-O	59 421.0	3.21570×10^{-1}	0.01738	negligible
EmoClassRFC-E	EmoClassCodeBERT-S	1226.0	5.21858×10^{-114}	−0.90604	large
EmoClassRFC-E	EmoClassCodeBERT-T	425.0	1.70137×10^{-115}	−0.91955	large
EmoClassRFC-E	EmoClassCodeBERT-E	0.0	2.70487×10^{-116}	−0.94291	large
EmoClassRFC-E	EmoClassCodeBERT-O	0.0	2.70488×10^{-116}	−0.94533	large
EmoClassRFC-O	EmoClassCodeBERT-S	1172.0	3.83828×10^{-114}	−0.90525	large
EmoClassRFC-O	EmoClassCodeBERT-T	417.0	1.48150×10^{-115}	−0.91886	large
EmoClassRFC-O	EmoClassCodeBERT-E	0.0	2.37949×10^{-116}	−0.94224	large
EmoClassRFC-O	EmoClassCodeBERT-O	0.0	2.37952×10^{-116}	−0.94465	large
EmoClassCodeBERT-S	EmoClassCodeBERT-T	29 327.0	4.90961×10^{-9}	−0.05782	negligible
EmoClassCodeBERT-S	EmoClassCodeBERT-E	10 957.0	3.83547×10^{-48}	−0.14031	negligible
EmoClassCodeBERT-S	EmoClassCodeBERT-O	8776.5	4.37258×10^{-57}	−0.15875	small
EmoClassCodeBERT-T	EmoClassCodeBERT-E	13 447.0	3.84252×10^{-31}	−0.08313	negligible
EmoClassCodeBERT-T	EmoClassCodeBERT-O	10 694.0	9.68241×10^{-42}	−0.10171	negligible
EmoClassCodeBERT-E	EmoClassCodeBERT-O	17 422.5	1.90539×10^{-6}	−0.02017	negligible

Results. Table 13 shows that the BERT-based and CodeBERT-based models performed for all emotion categories. We also notice that there is not much difference in the performance of models on augmented datasets and non-augmented datasets. These results show an interesting insight that data augmentation techniques need to be adapted using SE-specific vocabulary. In the future, we work on improving these data augmentation techniques using SE-specific data. Table 14 shows the results of Wilcoxon signed-rank and Cliff's delta. These results indicate that in most of the cases, the p-value obtained is lower than 0.05 and hence, the null hypothesis is rejected, which shows that the model performance is significantly different.

The BERT-based and CodeBERT-baed models perform best for emotion classification. There is no significant difference in the performance of models trained on augmented data and non-augmented data.

7. Threats to validity

This section delves into possible factors that could impact the credibility of this research. These factors are categorized into internal, external, and construct validity [66].

7.1. Internal validity

Internal validity relates to the potential design elements of a study that could affect its outcomes. One of those is overfitting, which emerges when a model, in its attempt to minimize loss, starts to memorize the training data rather than understand the underlying patterns [67] leading it to excel on the training dataset but underperform on unseen data. To prevent overfitting, the training process of all three models was monitored closely. Early stopping techniques and dropout were adopted for the LSTM model. Experiments were conducted with the BERT model to identify the most appropriate BERT pre-trained model and set of hyperparameters for our task. For the RFC model, the GridSearchCV function was used to obtain the best combination of parameters.

Data augmentation techniques, like word substitution and back translation, were employed to enrich our limited dataset. However, they pose challenges. Word substitution can change emotional nuances, and back translation might alter original sentiments, potentially misleading the model during training. To address these, we employed contextual word substitution and reviewed a subset of the augmented data for accuracy. For the back translation approach, the samples were increased once to avoid introducing duplicates. Furthermore, only a portion of the minority classes was increased using the augmentation methods. Additionally, labeling Stack Overflow posts can introduce interpretation variances. We countered this by using the gold label, determined by a majority vote system as noted by [19].

7.2. External validity

This section outlines potential limitations that might affect the broader applicability of the results from this study. This research utilized an annotated dataset extracted from Stack Overflow, a prominent Q&A platform for software developers. Although Stack Overflow is a major hub for developer discussions, it is worth noting that there are other platforms, such as GitHub, where developers also engage in conversations. Since the models implemented in this study were trained on a dataset extracted from Stack Overflow, the results might differ if evaluated on the dataset from another platform. Furthermore, factors such as differences in the domain, annotator biases, and varying annotation guidelines can impact performance as observed by [68]. Nonetheless, for a more comprehensive generalization, future research could incorporate posts and comments from other Q&A platforms.

7.3. Construct validity

Construct validity examines how well theoretical concepts are translated into actual observations. It evaluates whether the methods used in research truly capture the abstract ideas they intend to measure. Concerns about construct validity arise from the appropriateness of our evaluation metrics, and the reliability of the manually annotated dataset used. Evaluation metrics, including precision, recall, and the F_1-score, are the benchmarks against which the efficacy of sentiment analysis or emotion classification solutions are gauged, as supported by [2, 14, 40, 56]. These metrics serve as the foundation for understanding our

solutions' true performance and reliability. However, a significant aspect to consider is the source of our data. By using publicly available datasets utilized in prior works, we are not just leveraging the data but also inheriting any potential inaccuracies or biases inherent in the dataset. As a result, it is essential to recognize this inherited vulnerability when interpreting our findings and drawing conclusions.

8. Conclusion and future work

Detecting software engineers' emotions has become increasingly important in understanding team dynamics, improving collaboration, and enhancing overall productivity in software development projects. In this study, we implemented four different ML architectures – BERT, CodeBERT, RFC, and LSTM – for the emotion classification task. To improve the performance and robustness of the models, three techniques, word substitution, back translation, and Easy Data Augmentation, were used to augment the training data. Four variations of each architecture were implemented: one using data augmented through word substitution, the second using back-translated data, the third using easy data augmentation, and the fourth using the original data.

EmoClassBERT-substitution gives the highest F_1-score of 63%. The substitution method shows a considerable improvement as compared to the original models, for example, EmoClassLSTM-Substitution, EmoClassBERT-Substitution, EmoClassCodeBERT-Substitution, and EmoClassRFC-Substitution models show improvements of 13%, 5%, 5%, and 10% as compared to EmoClassLSTM-Original. Overall, the BERT-based and CodeBERT-based models perform best for emotion classification. The results reveal no significant difference in the performance of models trained on augmented data and non-augmented data.

This underscores the ability of transformer-based models to capture semantic and contextual relationships, making them particularly suited for tasks involving complex textual data such as emotion classification. In comparison, the LSTM models demonstrated inferior performance, likely due to limited data as they need abundant data for training. Despite being the simplest of the three models, the RFC provided better results than the LSTM model, highlighting the potential for traditional ML techniques in emotion classification tasks when combined with robust feature extraction and data augmentation. In summary, BERT mostly outperformed both RFC and LSTM in terms of the F_1-score. Furthermore, the augmentation techniques played a pivotal role in refining our models. Specifically, word substitution showcased a more pronounced improvement in the model's performance than back translation. Moreover, EmoClassBERT-Substitution demonstrated a reliable performance when compared to existing tools.

From the findings obtained, several interesting future directions are possible. For example, In the future, we plan to explore additional data augmentation techniques, such as Generative Adversarial Networks (GANs), and combine the strengths of various models to form an ensemble model. We plan to do an exhaustive comparison of various data augmentation techniques for emotion classification on the SE dataset as well as using a stack of data augmentation as proposed by [2]. We also plan to use LLM like RoBERT, and ALBERT to analyze their performance for emotion classification in the software engineering domain. Furthermore, deeper hyperparameter tuning could be performed. The dataset could also be expanded to encompass various sources, such as combining data from Stack Overflow and GitHub. Additionally, we will work on improving these data augmentation techniques using SE-specific data.

CRediT authorship contribution statement

Author 1: Methodology, software, investigation, writing – original draft, writing – review and editing, visualization.
Author 2: Conceptualization, software, writing – original draft, writing – review and editing, supervision, project administration.

Declaration of competing interest

No competing interest.

Funding

This research was conducted without any external funding.

References

[1] D. Girardi, F. Lanubile, N. Novielli, and A. Serebrenik, "Emotions and perceived productivity of software developers at the workplace," *IEEE Transactions on Software Engineering*, Vol. 48, No. 9, 2021, pp. 3326–3341.
[2] M.M. Imran, Y. Jain, P. Chatterjee, and K. Damevski, "Data augmentation for improving emotion recognition in software engineering communication," in *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, 2022, pp. 1–13.
[3] M. Ortu, B. Adams, G. Destefanis, P. Tourani, M. Marchesi et al., "Are bullies more productive? Empirical study of affectiveness vs. issue fixing time," in *12th Working Conference on Mining Software Repositories*. IEEE, 2015, pp. 303–313.
[4] S. Cagnoni, L. Cozzini, G. Lombardo, M. Mordonini, A. Poggi et al., "Emotion-based analysis of programming languages on Stack Overflow," *ICT Express*, Vol. 6, No. 3, 2020, pp. 238–242.
[5] N. Forsgren, M.A. Storey, C. Maddila, T. Zimmermann, B. Houck et al., "The space of developer productivity: There's more to it than you think." *Queue*, Vol. 19, No. 1, 2021, pp. 20–48.
[6] D. Girardi, N. Novielli, D. Fucci, and F. Lanubile, "Recognizing developers' emotions while programming," in *Proceedings of the ACM/IEEE 42nd international conference on software engineering*, 2020, pp. 666–677.
[7] D. Graziotin, F. Fagerholm, X. Wang, and P. Abrahamsson, "What happens when software developers are (un) happy," *Journal of Systems and Software*, Vol. 140, 2018, pp. 32–47.
[8] N. Novielli and A. Serebrenik, "Sentiment and emotion in software engineering," *IEEE Software*, Vol. 36, No. 5, 2019, pp. 6–23.
[9] B. Lin, N. Cassee, A. Serebrenik, G. Bavota, N. Novielli et al., "Opinion mining for software development: A systematic literature review," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, Vol. 31, No. 3, 2022, pp. 1–41.
[10] N. Imtiaz, J. Middleton, P. Girouard, and E. 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*, 2018, pp. 55–61.
[11] F. Calefato, F. Lanubile, F. Maiorano, and N. Novielli, "Sentiment polarity detection for software development," in *Proceedings of the 40th International Conference on Software Engineering*, 2018, p. 128.
[12] L. Yao and Y. Guan, "An improved LSTM structure for natural language processing," in *International Conference of Safety Produce Informatization (IICSPI)*. IEEE, 2018, pp. 565–569.

[13] J. Antony Vijay, H. Anwar Basha, and J. Arun Nehru, "A dynamic approach for detecting the fake news using random forest classifier and NLP," in *Computational Methods and Data Engineering*. Springer, 2020, pp. 331–341.
[14] D. Bleyl and E.K. Buxton, "Emotion recognition on Stack Overflow posts using BERT," in *International Conference on Big Data (Big Data)*. IEEE, 2022, pp. 5881–5885.
[15] T. Ahmed, A. Bosu, A. Iqbal, and S. Rahimi, "SentiCR: A customized sentiment analysis tool for code review interactions," in *32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2017, pp. 106–111.
[16] V. Kumar, A. Choudhary, and E. Cho, "Data augmentation using pre-trained transformer models," *arXiv preprint arXiv:2003.02245*, 2020.
[17] S. Shleifer, "Low resource text classification with ulmfit and backtranslation," *CoRR*, Vol. abs/1903.09244, 2019. [Online]. http://arxiv.org/abs/1903.09244
[18] A. Koufakou, D. Grisales, O. Fox et al., "Data augmentation for emotion detection in small imbalanced text data," *arXiv preprint arXiv:2310.17015*, 2023.
[19] 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*, MSR '18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 14–17. [Online]. https://doi.org/10.1145/3196398.3196453
[20] 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, 2017, pp. 521–564.
[21] B. Liu, *Sentiment analysis and opinion mining*. Springer Nature, 2022.
[22] P. Sudhir and V.D. Suresh, "Comparative study of various approaches, applications and classifiers for sentiment analysis," *Global Transitions Proceedings*, Vol. 2, No. 2, 2021, pp. 205–211.
[23] O. Bruna, H. Avetisyan, and J. Holub, "Emotion models for textual emotion classification," *Journal of Physics: Conference Series*, Vol. 772, No. 1, 2016, p. 012063. [Online]. https://dx.doi.org/10.1088/1742-6596/772/1/012063
[24] Z. Teng, F. Ren, and S. Kuroiwa, "Retracted: recognition of emotion with svms," in *Computational Intelligence: International Conference on Intelligent Computing*. Springer, 2006, pp. 701–710.
[25] E. Guzman and B. Bruegge, "Towards emotional awareness in software development teams," in *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2013. New York, NY, USA: Association for Computing Machinery, 2013, pp. 671–674. [Online]. https://doi.org/10.1145/2491411.2494578
[26] A. Fontão, O.M. Ekwoge, R. Santos, and A.C. Dias-Neto, "Facing up the primary emotions in mobile software ecosystems from developer experience," in *Proceedings of the 2nd Workshop on Social, Human, and Economic Aspects of Software*, WASHES '17. New York, NY, USA: Association for Computing Machinery, 2017, pp. 5–11. [Online]. https://doi.org/10.1145/3098322.3098325
[27] E. Guzman, D. Azócar, and Y. Li, "Sentiment analysis of commit comments in github: An empirical study," in *Proceedings of the 11th Working Conference on Mining Software Repositories*, MSR 2014. New York, NY, USA: Association for Computing Machinery, 2014, pp. 352–355. [Online]. https://doi.org/10.1145/2597073.2597118
[28] A. Murgia, P. Tourani, B. Adams, and M. Ortu, "Do developers feel emotions? an exploratory analysis of emotions in software artifacts," in *Proceedings of the 11th Working Conference on Mining Software Repositories*, MSR 2014. New York, NY, USA: Association for Computing Machinery, 2014, pp. 262–271. [Online]. https://doi.org/10.1145/2597073.2597086
[29] G. Uddin and F. Khomh, "Automatic mining of opinions expressed about apis in stack overflow," *IEEE Transactions on Software Engineering*, Vol. 47, No. 3, 2019, pp. 522–559.
[30] A. Ciurumelea, A. Schaufelbühl, S. Panichella, and H.C. Gall, "Analyzing reviews and code of mobile apps for better release planning," in *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2017, pp. 91–102.
[31] X. Gu and S. Kim, "" what parts of your apps are loved by users?"(t)," in *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2015, pp. 760–770.

[32] S. Panichella, A. Di Sorbo, E. Guzman, C.A. Visaggio, G. Canfora et al., "How can i improve my app? classifying user reviews for software maintenance and evolution," in *2015 IEEE international conference on software maintenance and evolution (ICSME)*. IEEE, 2015, pp. 281–290.

[33] M.M. Rahman, C.K. Roy, and I. Keivanloo, "Recommending insightful comments for source code using crowdsourced knowledge," in *2015 IEEE 15th International Working Conference on Source Code Analysis and Manipulation (SCAM)*. IEEE, 2015, pp. 81–90.

[34] R. Jongeling, S. Datta, and A. Serebrenik, "Choosing your weapons: On sentiment analysis tools for software engineering research," in *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2015, pp. 531–535.

[35] M. Thelwall, K. Buckley, G. Paltoglou, A. Kappas, and D. Cai, "Sentiment strength detection in short informal text," *Journal of the American Society for Information Science and Technology*, 2010.

[36] E. Loper and S. Bird, "Nltk: The natural language toolkit," in *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, ETMTNLP '02, Vol. 1. Association for Computational Linguistics, 2002, pp. 63–70. [Online]. https://doi.org/10.3115/1118108.1118117

[37] M.R. Islam and M.F. Zibran, "Leveraging automated sentiment analysis in software engineering," in *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, 2017, pp. 203–214.

[38] F. Calefato, F. Lanubile, F. Maiorano, and N. Novielli, "Sentiment polarity detection for software development," *Empirical Software Engineering*, Vol. 23, No. 3, 2017, pp. 1352–1382.

[39] D. Graziotin, X. Wang, and P. Abrahamsson, "Do feelings matter? on the correlation of affects and the self-assessed productivity in software engineering," *Journal of Software: Evolution and Process*, Vol. 27, No. 7, 2015, pp. 467–487. [Online]. https://onlinelibrary.wiley.com/doi/abs/10.1002/smr.1673

[40] F. Calefato, F. Lanubile, and N. Novielli, "Emotxt: A toolkit for emotion recognition from text," *CoRR*, Vol. abs/1708.03892, 2017. [Online]. http://arxiv.org/abs/1708.03892

[41] N. Boucher, I. Shumailov, R.J. Anderson, and N. Papernot, "Bad characters: Imperceptible NLP attacks," *CoRR*, Vol. abs/2106.09898, 2021. [Online]. https://arxiv.org/abs/2106.09898

[42] Y. HaCohen-Kerner, D. Miller, and Y. Yigal, "The influence of preprocessing on text classification using a bag-of-words representation." *PloS one*, Vol. 15, 2020, p. e0232525.

[43] J.J. Webster and C. Kit, "Tokenization as the initial phase in NLP," in *The 14th international conference on computational linguistics*, 1992.

[44] N. Rahimi, F. Eassa, and L. Elrefaei, "An ensemble machine learning technique for functional requirement classification," *symmetry*, Vol. 12, No. 10, 2020, p. 1601.

[45] A. Humphreys and R.J.H. Wang, "Automated text analysis for consumer research," *Journal of Consumer Research*, Vol. 44, 2018, pp. 1274–1306. [Online]. https://api.semanticscholar.org/CorpusID:168854843

[46] L. Tian, C. Lai, and J.D. Moore, "Polarity and intensity: The two aspects of sentiment analysis," *arXiv preprint arXiv:1807.01466*, 2018.

[47] A. Ali, S.M. Shamsuddin, and A.L. Ralescu, "Classification with class imbalance problem," *Int. J. Advance Soft Compu. Appl*, Vol. 5, No. 3, 2013, pp. 176–204.

[48] X.Y. Liu, J. Wu, and Z.H. Zhou, "Exploratory undersampling for class-imbalance learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, Vol. 39, No. 2, 2009, pp. 539–550.

[49] Q. Liu, M.J. Kusner, and P. Blunsom, "A survey on contextual embeddings," 2020.

[50] P. Bahad, P. Saxena, and R. Kamal, "Fake news detection using bi-directional lstm-recurrent neural network," *Procedia Computer Science*, Vol. 165, 2019, pp. 74–82, 2nd International Conference on Recent Trends in Advanced Computing DISRUP-TIV INNOVATION. [Online]. https://www.sciencedirect.com/science/article/pii/S1877050920300806

[51] C. Zhou, C. Sun, Z. Liu, and F.C.M. Lau, "A C-LSTM neural network for text classification," *CoRR*, Vol. abs/1511.08630, 2015. [Online]. http://arxiv.org/abs/1511.08630

[52] Y. Zhang, "Research on text classification method based on lstm neural network model," in *2021 IEEE Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC)*, 2021, pp. 1019–1022.

[53] R. Adarsh, A. Patil, S. Rayar, and K. Veena, "Comparison of VADER and LSTM for sentiment analysis," *International Journal of Recent Technology and Engineering*, Vol. 7, No. 6, Mar. 2019, pp. 540–543.

[54] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, Vol. 9, No. 8, 11 1997, pp. 1735–1780. [Online]. https://doi.org/10.1162/neco.1997.9.8.1735

[55] B. Lindemann, T. Müller, H. Vietz, N. Jazdi, and M. Weyrich, "A survey on long short-term memory networks for time series prediction," *Procedia CIRP*, Vol. 99, 2021, pp. 650–655, 14th CIRP Conference on Intelligent Computation in Manufacturing Engineering, 15–17 July 2020. [Online]. https://www.sciencedirect.com/science/article/pii/S2212827121003796

[56] H. Batra, N.S. Punn, S.K. Sonbhadra, and S. Agarwal, "BERT-based sentiment analysis: A software engineering perspective," in *Database and Expert Systems Applications*. Springer International Publishing, 2021, pp. 138–148.

[57] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, Vol. abs/1810.04805, 2018. [Online]. http://arxiv.org/abs/1810.04805

[58] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones et al., "Attention is all you need," *CoRR*, Vol. abs/1706.03762, 2017. [Online]. http://arxiv.org/abs/1706.03762

[59] Y. Al Amrani, M. Lazaar, and K.E. El Kadiri, "Random Forest and Support Vector Machine based hybrid approach to sentiment analysis," *Procedia Computer Science*, Vol. 127, 2018, pp. 511–520, proceedings of the first International Conference On Intelligent Computing in Data Sciences, ICDS2017. [Online]. https://www.sciencedirect.com/science/article/pii/S1877050918301625

[60] L. Breiman, "Random forests," *Machine Learning*, Vol. 45, No. 1, 2001, pp. 5–32. [Online]. https://doi.org/10.1023/A:1010933404324

[61] M. Wu, Y. Yang, H. Wang, and Y. Xu, "A deep learning method to more accurately recall known lysine acetylation sites," *BMC Bioinformatics*, Vol. 20, No. 1, 2019, p. 49. [Online]. https://doi.org/10.1186/s12859-019-2632-9

[62] Z. Feng, D. Guo, D. Tang, N. Duan, X. Feng et al., "Codebert: A pre-trained model for programming and natural languages," *arXiv preprint arXiv:2002.08155*, 2020.

[63] J. Ramos et al., "Using tf-idf to determine word relevance in document queries," in *Proceedings of the first instructional conference on machine learning*. Citeseer, 2003, pp. 29–48.

[64] C. Tantithamthavorn, S. McIntosh, A.E. Hassan, and K. Matsumoto, "An empirical comparison of model validation techniques for defect prediction models," *IEEE Transactions on Software Engineering*, Vol. 43, No. 1, 2016, pp. 1–18.

[65] J. Romano, J.D. Kromrey, J. Coraggio, and J. Skowronek, "Appropriate statistics for ordinal level data: Should we really be using t-test and cohen'sd for evaluatng group differences on the nsse and other surveys," in *annual meeting of the Florida Association of Institutional Research*, Vol. 177, 2006, p. 34.

[66] L. Baldwin, "Internal and external validity and threats to validity," in *Research concepts for the practitioner of educational leadership*. Brill, 2018, pp. 31–36.

[67] X. Ying, "An overview of overfitting and its solutions," *Journal of Physics: Conference Series*, Vol. 1168, No. 2, feb 2019, p. 022022. [Online]. https://dx.doi.org/10.1088/1742-6596/1168/2/022022

[68] L.A. Cabrera-Diego, N. Bessis, and I. Korkontzelos, "Classifying emotions in Stack Overflow and JIRA using a multi-label approach," *Knowledge-Based Systems*, Vol. 195, 2020, p. 105633. [Online]. https://www.sciencedirect.com/science/article/pii/S0950705120300939

Authors and affiliations

Didi Awovi Ahavi-Tete
e-mail: didi.ahavitete@gmail.com
ORCID: https://orcid.org/0009-0001-7348-2763
School of Computer Science and Mathematics,
Keele University, United Kingdom

Sangeeta Sangeeta
e-mail: s.sangeeta@keele.ac.uk
ORCID: https://orcid.org/0000-0002-3734-7871
School of Computer Science and Mathematics,
Keele University, United Kingdom