Sentiment Analysis for Software Engineering; A Study on the Effectiveness of Data Augmentation and Ensembling using Transformer-based Models

Zubair Rahman Tusar
Computer Science and Engineering
Islamic University of Technology
Gazipur, Bangladesh
zubairrahman@iut-dhaka.edu

Sadat Bin Sharfuddin
Computer Science and Engineering
Islamic University of Technology
Gazipur, Bangladesh
sadatsharfuddin@iut-dhaka.edu

Muhtasim Abid
Computer Science and Engineering
Islamic University of Technology
Gazipur, Bangladesh
muhtasimabid@iut-dhaka.edu

***Abstract*—Sentiment analysis for software engineering has undergone numerous research to efficiently develop tools and approaches for software engineering contents. State-of-the art tools achieved better performance using transformer-based models like BERT, RoBERTa to classify sentiment polarity. However, existing tools overlooked data imbalance problem and did not consider the efficiency of ensembleing multiple pre-trained models on sSE-specific datasets.**

However, existing tools overlooked data imbalance problem and did not consider the efficiency of ensembleing multiple pre-trained models on SE-specific datasets. To overcome those limitations, we use context specific data augmentation using SE-specific vocabulary and ensembled multiple models to classify sentiment polarity. Using four gold-standard SE-specific datasets, we trained our ensembled models and evaluated the performance over the individual pre-trained transformer-based models.

Our approach achieved highest (%) scores improvement on weighted average F1 scores and macro-average F1 scores. Our results show that the ensemble models outperform the pre-trained transformer-based models on the original datasets and that data augmentation further improve the performance of all the previous approaches.

***Index Terms*—Sentiment Analysis, Pre-Trained Transformer-based Models, Ensembling, Data Augmentation**

I. Introduction

Sentiment analysis is a computational analysis of people's attitudes, emotions, and views regarding an entity, which might be a person, an event, or perhaps a topic [1]. It can be used to determine the emotional tone of a body of writing. For a given text unit, it can determine whether the text expresses a positive, negative, or neutral sentiment.

In recent times, the software engineering community has begun using sentiment analysis and its tools for various purposes. It has been used to investigate the role of emotions in IT projects and software development [2] [3], finding relations between developers' sentiment and software bugs [4], analyzing the relation between sentiment, emotions, and politeness of developers with the time to fix a Jira issue [5], and so on. Since software development relies on human efforts and interactions, it is more susceptible to the practitioners' emotions. All these research works make use of sentiment analysis to analyze various aspects of the software engineering domain that are affected by sentiment and emotion and how they can be used to improve this domain overall.

There are a number of tools available for sentiment analysis that are being used in software engineering. These tools and techniques mainly follow three types of approaches: unsupervised approach, supervised approach, and Transformer-based approach.

Unsupervised approach or the lexicon-based approach are used by tools like SentiStrength-SE [6] and DEVA [7]. These tools achieved performance improvement by considering SE-specific texts. The reason behind this is that the technical terms that are specific for a domain have differences in meaning. Next, there are tools using supervised learning approach. Examples of such tools are SentiCR [8] and Senti4SD [9]. SentiCR is trained on code review comments. It classifies code review comments into two classes - negative and non-negative. Senti4SD is trained on a gold standard of Stack Overflow questions, answers, and comments. It is trained for supporting sentiment analysis in developers' communication channels. However, in recent times, these approaches somewhat subsided due to the proliferation of transformer model-based approaches.

The transformer model-based approaches for sentiment analysis have gained much attention in recent times. Fine-tuning different variants of pre-trained transformer models like BERT, RoBERTa, and XLNet for software engineering-specific content can outperform existing tools [10]. The pre-trained transformer-based models have been shown to outperform the unsupervised and supervised tools even when they are trained and validated by software engineering specific content. As shown in [10], pre-trained transformer models outperformed the existing best-performing tools, namely, SentiStrength, SentiStrength-SE, SentiCR, Senti4SD, and Stanford CoreNLP by 6.5% to 35.6% in terms of macro and micro-averaged F1-scores. Although the pre-trained transformer models perform better than the previous approaches, there is room for improvement as they suffer from issues like small dataset size, and class imbalance in the datasets used for fine tuning the models.

[you can make 2 lines about these problems].

To solve these problems, we conduct a study on the effectiveness of text augmentation and ensembling on sentiment analysis using transformer-based models.

We evaluated the proposed approach using four publicly available datasets with annotated sentiment polarities, each of which is a gold standard dataset. We chose three pre-trained transformer-based models: BERT, RoBERTa, and XLNet. We apply text augmentation with SE-specific Word2Vec [11] and EDA (Easy Data Augmentation) [12]. We use two versions of augmented datasets. For the first one, we do not consider the class imbalance issue and augment the datasets equally for all the classes, and for the second one, which we call controlled augmentation, we consider the class imbalance issue and apply augmentation in a way to balance the classes along with increasing the dataset size. Then on the augmented datasets, we perform sentiment analysis by ensembling the chosen transformer-based models and analyze the results to observe performance improvement over contemporary approaches. Specifically, we aim to investigate the following research questions.

- **RQ1**: Do the ensemble models outperform the pre-trained models on the original datasets?
- **RQ2**: Does data augmentation improve the performances of the models?

The experimental results demonstrate that the ensemble models outperform all the pre-trained transformer models in three out of the four original datasets, i.e., datasets without any augmentation, in terms of weighted- and macro-average F1 scores. The results further demonstrate that the augmentation approaches aid the performance of all the pre-trained transformer model approaches as well as the ensemble models in two out of the three datasets in terms of weighted and macro average F1 scores.

The main contributions of this paper are:

- Use of stacking ensemble on pre-trained transformer-based models to improve SE-specific sentiment analysis.
- Controlled augmentation for tackling class imbalance in the datasets in order to improve effectiveness of transformer based models on SE-specific sentiment analysis.
- Use of SE-based Word2Vec for data augmentation, which is a novel approach.

Structure of the paper: Section II talks about the background study done for this work. Section III introduces the related works. Section IV demonstrates the methodology adopted for our research work. Here we talk about the datasets, transformer-based models, ensemble, and augmentation approaches we use. Next in section V, we evaluate, analyze, and discuss the results of our experiments and present the main findings of the research questions. Then we analyze the limitations of our work in section VI. Finally, section VII presents the future work and concludes the paper.

II. Background Study

In prior studies on sentiment analysis, psychologists explored two different approaches to deciding upon the sentiment of a text. One method leverages a two-dimensional model. The two axes of this bi-dimensional model are valence (pleasing vs. unpleasing) and valence intensity. The second approach assumes that a limited set of emotions exists, but there is no consensus about the number of these emotions or their nature. [13] [14] The initial sentiment analysis process consisted of manual mining of human opinions through interviews, surveys, etc. [15] The noticeable limitation to such an approach was that the actual sentiment may be biased or suppressed due to different numbers of variables. [16] The researchers were interested in automated ways to determine emotions to solve this. The computerized approach can be grouped into the supervised or unsupervised method from a broader perspective. The supervised process required a labeled dataset, meaning a dataset consisting of samples with corresponding sentiment labels. To label these texts, researchers have explored both manual and automated techniques. However, the high cost, chances of prohibition, and biases in labeling were evident. In the case of unsupervised approaches, a dataset was not required, but the problem of being sensitive to the domain words remained.

A. Sentiment analysis for Software Engineering

Emotions are an intrinsic element of human nature and can directly affect the task, team play, creativity, etc. Most software developments are a collaborative effort consisting of the collective contribution of a diverse team. Because software development is so reliant on human action and the interrelation of the whole team, it is more vulnerable to individual emotions. Consequently, a thorough understanding of developers' emotions and the elements that influence them may be used to facilitate better collaborations, work assignments, and the development of measures to improve job satisfaction, which can lead to greater productivity. The use cases of sentiment analysis in software engineering studies are not only limited to the factors mentioned earlier. Researchers have also explored a way to recommend software packages by leveraging the sentiment of commit comments on GitHub projects. A number of researches have been conducted in this regard. The prior ones leveraged the off-the-shelf sentiment analysis tools. The problem with these approaches was their sensitivity to the domain. [9] Thus, researchers are inclined to use specific techniques for software engineering.

B. Unsupervised approaches

The most prior ones explored by the researchers, the off-the-shelf sentiment analysis tools not designed/trained for the software engineering domain, were NLTK, Stanford core NLP, sentistrength, etc. As these tools were designed/trained in a different context, the researchers came to a consensus of poor performance based on the measure of the accuracy of detecting the sentiment of a given text. So, the following line of work in the community was driven towards designing a solution specific to the software engineering domain. Upon achieving poor accuracy with sentistrength, Islam and Zibran et al. worked on the dictionary that the tool leverages to include software engineering specific words and update the existing polarity words based on the context. The authors further expanded their work to add the capability to detect emotions other than positive, negative, and neutral to their approach and offered the community DEVA, a tool for multiple sentiment detection.

C. Supervised approaches

Stanford Core NLP was evaluated on software engineering texts by Lin et al. [17] which did not perform well as the model was trained on movie reviews. Researchers afterward used the Gradient Boosting Tree (GBT) based approach named SentiCR, which leverages the bag of words extracted from the training dataset. The authors additionally adopted the "Negation" technique. In a later work, the authors proposed a method that also used a supervised method combining different features to determine the sentiment of a given text. These approaches outperformed the previously mentioned unsupervised approaches.

D. Utilization of Pretrained models

Siba M et al. explored the word2vec approach for software engineering texts and found the significant differences for exact words in natural language vs. software engineering context. In the Natural Language Processing (NLP) community, the transformer-based models were introduced, outperforming existing approaches in different tasks, including text classification. In this line of work, T. Zhang et al. [10] fine-tuned and evaluated transformer-based models on software engineering domain-specific texts. This experiment provided the community with the possibility as the results outperformed the existing unsupervised and supervised software engineering-specific sentiment analysis tools. In another work, Batra et al [18] author experimented with different transformer-based models and a boosting-based ensembling technique in text sentiment analysis in the software engineering domain. This approach also displayed improvement compared to existing tools.

E. Data Augmentation approaches

However, the class imbalance was evident in the datasets used in these approaches, meaning the number of samples for different sentiment polarities, i.e., positive, negative, and neutral, differed by a large margin. This issue causes supervised approaches to perform poorly. There are several strategies to tackle this issue in the community. One line of work is the Easy Data Augmentation (EDA) approach proposed by Wei et al. [12]. The authors evaluated this technique and found that the improvement rate of this technique increases inversely proportional to the size of the dataset.

III. Related works

In this subsection, we provide a brief description of the related works. The related work can be divided into four groups: unsupervised approaches for SE-based sentiment analysis, Supervised approaches for SE-based sentiment analysis, pre-trained models of NLP for sentiment analysis, and the Ensembling of pre-trained models.

A. Unsupervised approaches for SE based sentiment analysis

1) Senti Strength:: Thelwall et al. [19] developed the lexicon-based sentiment analysis technique named SentiStrength, which uses dictionaries of both formal terms and informal texts (like - slang, emoticon). Each and every word included in the dictionaries are assigned with a specific sentiment strength. SentiStrength categorizes sentences into positive and negative emotions and determines the strength of the emotions based on the dictionaries and linguistic analysis.

2) Senti Strength SE:: Islam and Zibran analyzed 151 Jira issue comments using SentiStrength, which produces wrong outputs. Investigating the reasons for less accuracy of SentiStrength, they found 12 reasons, of which domain-specific meanings of words were most prevalent. So they built a modified version of SentiStrength by adding a domain-specific dictionary [16]. New sentiment words and negations were added to the dictionary. It's the first sentiment analysis tool where SE-specific context was considered.

B. Supervised approaches for SE based sentiment analysis

1) Stanford CoreNLP: Socher et al [20] introduced Stanford CoreNLP for single-sentence sentiment classification; polarity along with the sentiment value of a sentence is returned by the tool. Stanford CoreNLP is trained with the Recursive Neural Tensor Network proposed by Socher et al. on the Stanford Sentiment Treebank.

2) SentiCR: Ahmed et al. developed SentiCR [8] specifically for code review comments. It classifies code review comments into two classes - negative and non-negative. The supervised classifier used in sentiCR is GBT(Gradient Boosting Tree) [21], as it gave the highest precision, recall, and accuracy among the eight evaluated classifiers.

3) Senti4SD: It's the first supervised learning-based tool to generate feature vectors. Senti4SD [9] makes use of three features - SentiStrength lexicons, n-gram extracted keywords from the dataset, Word representations in a distributional semantic model(DSM) exclusively trained on StackOverflow data. Four prototype vectors are calculated; p_pos, p_neg, p_neu is the sum of positive polarity word vectors, negative polarity word vectors, and neutral polarity word vectors, respectively, and p_subj is the sum of p_pos and p_neg. Semantic features are extracted from these four vectors. Finally, using Support Vector Machines, Senti4SD is trained to identify sentiment polarities of text based on mentioned features.

C. Pre-trained models of NLP for sentiment analysis

1) BERT: A deep learning model designed to learn contextual word representations from unlabeled texts [22]. It's based on the transformer architecture but doesn't have the decoder, rather contains only a multi-layer bidirectional transformer encoder. The pre-training of the model is accomplished by optimizing two tasks - masked language modeling (MLM) and next sentence prediction(NSP). Originally there were two

TABLE I: Datasets

dataset	total samples	positive samples	neutral samples	negative samples
App review	341	186	25	130
Stack overflow	1,500	131	1,191	178
Jira	926	290		636
Github	7,122	2,013	3,022	2,087

implementations of BERT: BERTBase with 12 layers, 12 self-attention heads, 110M parameters, and a hidden layer size of 768 and BERTLarge with 24 layers, 16 self-attention heads, 340M parameters, and 1024 hidden layer size. Our work uses BERTBase.

2) RoBERTa(Robustly optimized BERT approach): A modified version of BERT that changed pre-training steps by using larger mini-batch sizes to train over more data for a huge time, train on longer sequences to remove NSP loss and train with dynamic masking. When Liu et al [23] released it, it had achieved state - of - the art results on GLUE, RACE and SQuAD benchmarks, surpassing BERT.

3) XLNet: [24] Based on Transformer-XL, it uses segment recurrence mechanism and relative encoding. To address the individual weakness of autoregressive language modeling(AR) and autoencoding(AE) , it combines their strengths. XLNet performs better than BERT on 20 tasks, including sentiment analysis, especially for long texts.

D. Ensembling of pre-trained models

Batra et al. [18] used data augmentation through lexical-based substitution and back translation; as a pre-processing step to help train and fine-tune BERT variants - BERT, RoBERTa, and ALBERT. Then a weighted voted scheme was applied to the final Softmax layer output of the BERT variants to ensemble the models and achieve the final weighted prediction.

IV. Methodology

Initially, to compare our methodologies, we replicated one unsupervised approach, Sentistrength-SE, proposed by Islam and Zibran [16], and one supervised approach, SentiCR, proposed by Ahmed et al. [8]. An overview of our methodology is shown in Figure 1.

A. Dataset

We used 4 gold-standard datasets with annotated sentiment polarity for our study.

1) : Jira issues [17] The original dataset by Ortu et al. [5] had four labels of emotions : love, joy, anger and sadness which Lin et al. [17] brought down to two labels by annotating sentences with love and joy with positive polarity and sentences with anger and sadness with negative polarity.

2) App reviews: Villaroel et al. [25] presented 3000 reviews from which 341 reviews were randomly selected by Lin et al. [17]. The dataset has 5% confidence interval and 95% confidence level making it statistically significant. In this dataset the ratio of four categories of reviews - request for improving non-functional requirements, suggestion of new features, bug reporting and other were maintained.

3) Stack Overflow posts (SO): There are 1500 sentences in total. This dataset was gathered by Lin et al. [17] from a July 2017 Stack Overflow dump. They choose threads that are (i) labeled with Java and (ii) include one of the terms library, libraries, or API (s). After that, they chose 1,500 words at random and classified their sentiment polarities manually.

4) GitHub: The dataset has 7,122 sentences extracted from GitHub commit comments and pull-requests. An iterative extraction was performed on the dataset of Pletea et al. [26] by Novielli et al. [27] to obtain annotated text units.

TABLE II: Pre-trained transformer models

Architecture	Used Model	Parameters	Layers	Hidden	Heads
BERT	bert-base-uncased	110M	12	768	12
RoBERTa	roberta-base	120M	12	768	12
XLNet	xlnet-base-cased	110M	12	768	12

B. Fine-tuning transformer-based models

The first step in our research methodology was to fine-tune already pre-trained transformer-based models for the down-stream tasks. To do this, we split each dataset under our study and kept 70% for the train-set and 30% for the test-set. We fine-tuned three transformer-based models separately for each dataset with the train sets with the following configuration. The models are BERT, RoBERTa and XLNet. We refer to these models collectively as pre-trained transformers (PTT). The following hyper-parameter values for the BERT fine-tuning technique, according to the authors' paper[13], operate well across all tasks: (1) Batch size: 16, 32; (2) Number of epochs: 2, 3, 4; (3) Learning rate (Adam): 5e-5, 3e-5, 2e-5; We run all of these models in two epochs with a batch size of 16. We also used the learning rate of 2e-5 and utilized the AdamW optimizer. [] lists the models in the Hugging-face Transformers collection, along with their names and default configurations.

C. Ensemble

Researchers have studied that merging numerous models can result in a more robust model. We explored the stacking-based ensemble technique to aggregate the performance of the different transformer-based models. Because each of the models (that are part of the ensemble) has a distinct basic functioning, the range of predictions is substantially broader in the case of the ensemble. Each model was pre-trained on a particular language modeling task, such as BERT's next sentence prediction, XLNet's auto-regressive approach, and RoBERTa's dynamic masking. From this intuition, we take the softmax outputs of the last layer of these models for the text under study and use them as features to aggregate. We experimented with two different approaches to aggregate these features and compared the results. In one approach, we adopted a decision tree-based approach and used Random Forest (RF), and in another one, we adopted a regression-based approach and used Logistic Regression (LR). Three of the datasets (SO, App review, and Github)under our study have three classes. So for the logistic regression approach for these datasets, we adopted the one-vs-rest (OvR) method. Thus, this final aggregating model produces the final prediction.

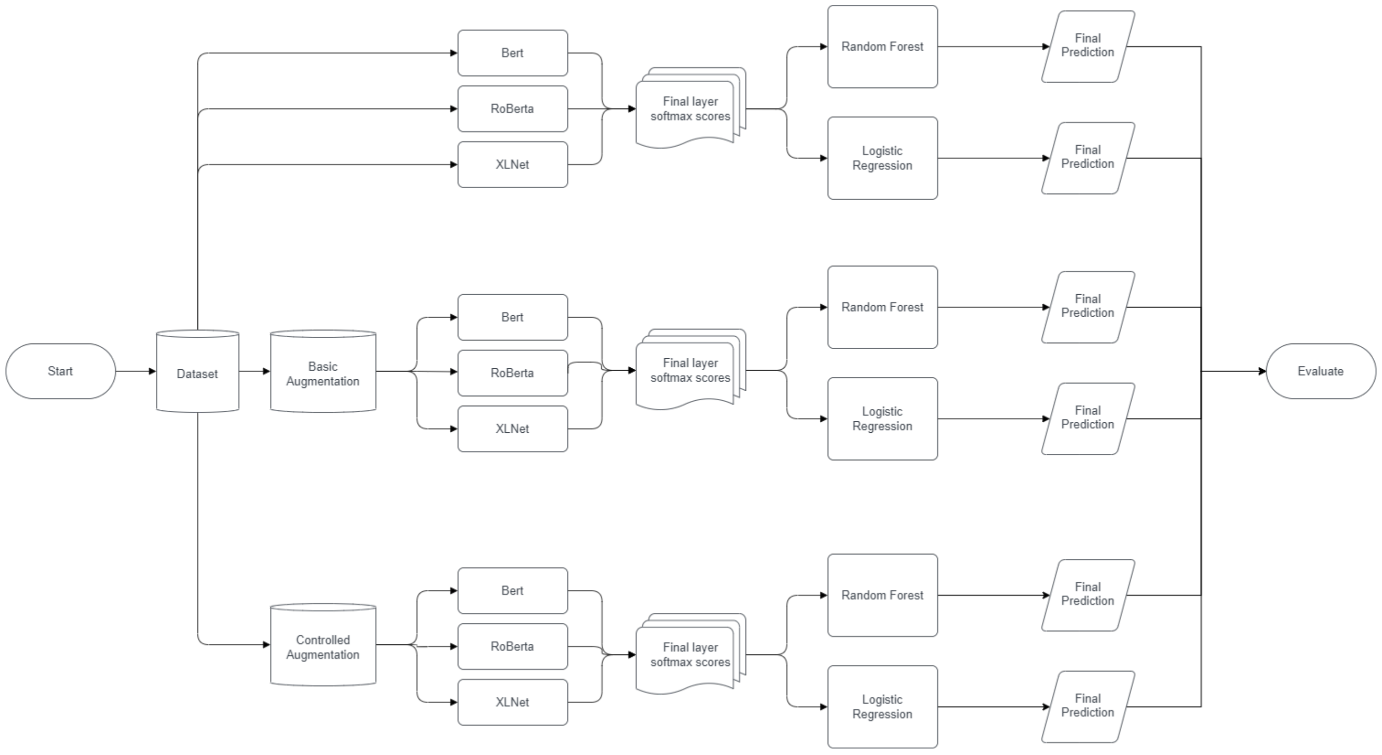
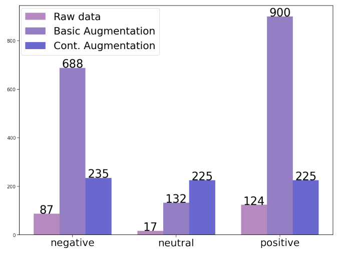
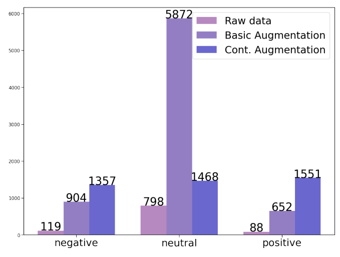


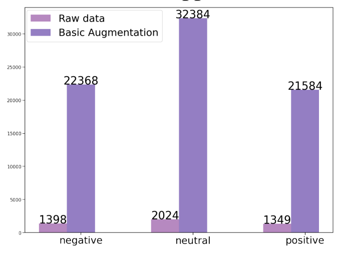
Fig. 1: Overview of the methodology



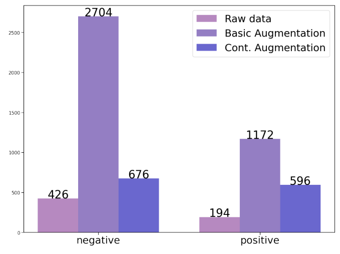
(a) Dataset: App Reviews



(b) Dataset: Stack Overflow



(c) Dataset: Github



(d) Dataset: Jira

Fig. 2: Class frequency distributions for datasets under study

D. Augmentation

It is evident from the figure [] that three of the four datasets (App reviews, SO, and Jira) have a small number of samples and a significant class imbalance issue. To aid the small dataset size, we adopted three techniques from the work of [] authors, namely Random synonym replacement: To replace random words in a sentence with their synonyms; Random deletion: To delete random words in a sentence; Random insertion: To insert random words in the sentence. For random synonym replacement, we applied two different techniques. One was to replace randomly selected words with the synonyms found in Natural Language Toolkit (NLTK) wordnet library. Another one is to replace using most similar words provided by a word2vec model trained specifically on software engineering text as an artifact of the study conducted by Siba m et al. []. As the Github dataset has more than 4k samples in the train-set, we do not apply augmentation on this. Our first approach was to tackle the issue of the dataset being small in size. Thus we applied the data augmentation techniques mentioned above to all the sample texts in the SO, Jira, and App reviews data sets regardless of the class distribution. We refer to this step as Basic Augmentation throughout the paper. In our second experiment, we applied the data augmentation techniques to balance the class-wise frequency distribution. For each dataset, we augmented texts for the class with a size smaller than the class with the maximum size until the distribution was reasonably balanced. We refer to this step as Controlled Augmentation throughout the paper.

V. Evaluation and Discussion

In this section, we report, compare and analyze the performance of the pre-trained transformer-based models and the ensemble models on the four datasets described in Chapter IV. We conducted augmentation on all the datasets except for the GH dataset. For each dataset, we report the performance of the approaches we used on the original version, the basic augmented version, and the controlled augmented version. For comparison, we also show the performance of SentiStrength-SE as a representative of the lexicon approach-based sentiment analysis tool and SentiCR as a representative of the supervised approach-based sentiment analysis tool only on the original version of the datasets. We highlight the best performance in terms of the two main metrics (i.e., weighted-average F1 scores and macro-average F1 scores) in bold. We answer the research questions based on the experimental results as follows.

*A. **RQ1: Do the ensemble models outperform the pre-trained models on the original datasets?***

We answer the RQ1 by analyzing the performance of ensemble models and pre-trained transformer-based models (PTT) only on the original version of the four datasets.

1) App-review dataset: The ensemble models outperform the PTT approaches in both weighted-average and macro-average F1 scores. The best performing PTT approach is BERT for both the F1 scores. BERT can achieve weighted- and macro-averaged F1 scores of .63 and .44, respectively. The RF-based ensemble model can achieve weighted- and macro-average F1 scores of .71 and .51, respectively, while the LR-based ensemble model can achieve weighted- and macro-averaged F1 scores of .69 and .49, respectively. Our results show that the RF-based ensemble model outperforms the best performing PTT approach by 8%-50% in terms of weighted-average F1 score by 7%-33% in terms of macro-average F1 score. This range is determined based on the improvement of the best and worst scores of the PTT approaches. The performance of all our approaches is relatively poor here because the size of this dataset is small, and the class distribution is highly imbalanced. But the ensemble models still perform better compared to the other approaches.

2) Stack Overflow dataset: The best performing PTT approach is RoBERTa for both weighted- and macro-average F1 scores for this dataset with an weighted- and macro-average F1 scores of .89 and .75, respectively, which is better than the performance achieved by the ensemble models. The LR-based ensemble model performs better than the RF-based and can achieve weighted- and macro-average F1 scores of .87 and .72, respectively. Although RoBERTa outperforms the ensemble models, the ensemble models outperform the other two PTT approaches. More specifically, the LR-based ensemble model outperforms the other two PTT approaches by 5%-12% in terms of weighted-average F1 score and by 10%-31% in terms of macro-average F1 score. This dataset also has a high imbalance in class distribution.

3) Jira dataset: The ensemble models outperform the PTT approaches in both weighted- and macro-average F1 scores here as well. The best performing PTT approach is RoBERTa which can achieve an F1 score of .95 for both weighted- and macro-average. Both the RF and LR-based ensemble models can achieve an F1 score of .96 for the weighted-average. But the macro-average F1 score of the RF-based ensemble model is .96, which is better than the .95 macro-average F1 score of the LR-based ensemble model. Our results show that the RF-based ensemble model outperforms the best performing PTT approach by 1%-6% in terms of weighted-average F1 score and by 1%-8% in terms of macro-average F1 score.

4) GitHub dataset: This dataset is the largest in size and also has the most balanced class distribution, which reflects in our results as well. All the approaches we use perform relatively well on this dataset. The ensemble models bring minor improvements to the already well-performing PTT approaches. Here, the best performing PTT approach is BERT which can achieve an F1 score of .91 for both weighted- and macro-average. The LR-based ensemble model also can achieve an F1 score of .91 for both weighted- and macro-average. But the RF-based ensemble model outperforms them slightly. It can achieve an F1 score of .92 for both weighted- and macro-average. So the RF-based ensemble model outperforms the best performing PTT approach by 1%-2% for both weighted- and macro-average F1 scores.

RQ1 Main Findings:
The ensemble models outperform the pre-trained transformer-based models by a significant margin for three out of the four datasets. In the Stack Overflow dataset, RoBERTa, which is a PTT approach, performs better than the ensemble models.

*B. **RQ2: Does data augmentation improve the performances of the models?***

We answer the RQ2 by comparing and analyzing the performance of PTT and the ensemble models on the augmented versions of the datasets.

1) App-review dataset: In the basic augmentation approach, all the PTT and the ensemble approaches outperform the results achieved by training the models using the original train-sets in terms of both weighted-average and macro-average F1 scores. The best performance improvement is achieved for the bert-based-uncased PTT approach with a 23% improvement on weighted-average and macro-average F1 scores. RF ensemble approaches, however, achieve the best results when compared to PTT approaches, with weighted-average and macro-average F1 scores of .89 and .72, respectively. We observe the improvement for the LR ensemble model compared to when trained on the original train-set but do not achieve the best results. The noticeable improvement is for the neutral class. All the approaches fail to predict any neutral samples on the original dataset. Augmentation helped improve in this case, and we observed bert-base-uncased, and the RF ensemble approach improved in case of neutral sample detection. Overall, the results depict that data augmentation enhanced the performances of the models for this dataset.

We observe the best results for the controlled augmentation approach, especially for the classes with the least number of samples. In the controlled augmentation approach, all the PTT approaches achieve an improved weighted-average and macro-average F1 score. We observe that the LR ensemble approach achieves the best result with weighted-average and macro-

TABLE III: Results for the Appreview dataset

Dataset	Class	Negative			Neutral			Positive			Weighted-average			Macro-average		
	Model	P (%)	R (%)	F1 (%)	P (%)	R (%)	F1 (%)	P (%)	R (%)	F1 (%)	P (%)	R (%)	F1 (%)	P (%)	R (%)	F1 (%)
App Review	Sentistrength-SE	86	28	42	10	38	15	72	79	75	73	57	58	56	48	44
	SentiCR	77	79	78	14	12	13	84	84	84	76	77	77	58	58	58
	BERT	62	49	55	0	0	0	68	87	77	61	66	63	43	45	44
	RoBERTa	38	100	55	0	0	0	0	0	0	14	38	21	13	33	18
	XLNet	50	56	53	0	0	0	65	68	66	54	58	56	38	41	40
	Ensemble (RF)	64	86	73	0	0	0	84	74	79	70	73	71	49	53	51
	Ensemble (LR)	65	70	67	0	0	0	76	**82**	79	67	72	69	47	51	49
App Review Basic Augmentation	BERT	85	91	88	50	12	20	91	95	93	86	88	86	75	66	67
	RoBERTa	77	100	87	0	0	0	98	89	93	83	87	84	58	63	60
	XLNet	80	95	87	0	0	0	93	89	91	82	85	83	58	61	59
	Ensemble (RF)	87	95	91	40	25	31	95	94	94	88	**89**	**89**	74	71	72
	Ensemble (LR)	85	95	90	0	0	0	94	94	94	84	88	86	60	63	61
App Review Controlled Augmentation	BERT	86	88	87	43	38	40	92	92	92	86	87	87	74	73	73
	RoBERTa	83	91	87	33	12	18	94	95	94	85	88	86	70	66	66
	XLNet	95	86	90	33	25	29	87	95	91	86	87	86	72	69	70
	Ensemble (RF)	86	88	87	60	38	46	92	95	94	88	88	88	80	74	76
	Ensemble (LR)	84	88	86	80	50	62	94	95	94	89	89	**89**	86	78	**81**

TABLE IV: Results for the Stack Overflow dataset

Dataset	Class	Negative			Neutral			Positive			Weighted-average			Macro-average		
	Model	P (%)	R (%)	F1 (%)	P (%)	R (%)	F1 (%)	P (%)	R (%)	F1 (%)	P (%)	R (%)	F1 (%)	P (%)	R (%)	F1 (%)
Stack Overflow	Sentistrength-SE	38	14	20	82	92	87	29	23	26	72	77	74	50	43	44
	SentiCR	42	58	49	90	85	87	46	44	45	80	78	79	59	62	60
	BERT	69	42	53	86	97	91	80	28	41	83	84	82	78	56	62
	RoBERTa	82	71	76	91	97	94	86	42	56	89	89	**89**	86	70	75
	XLNet	80	20	32	81	99	89	0	0	0	74	81	75	54	40	41
	Ensemble (RF)	74	58	65	88	96	92	77	40	52	86	86	85	80	64	70
	Ensemble (LR)	77	63	69	89	97	93	81	40	53	87	88	87	82	66	72
Stack Overflow Basic Augmentation	BERT	67	54	60	90	95	93	73	56	63	86	87	86	77	68	72
	RoBERTa	67	69	68	92	92	92	59	56	57	86	86	86	73	73	73
	XLNet	75	68	71	91	95	93	73	56	63	88	88	88	80	73	**76**
	Ensemble (RF)	69	58	63	90	94	92	66	58	62	86	86	86	75	70	72
	Ensemble (LR)	70	64	67	91	94	93	69	56	62	87	87	87	77	72	74
Stack Overflow Controlled Augmentation	BERT	70	68	69	91	92	91	58	51	54	85	86	85	73	70	72
	RoBERTa	74	63	68	90	96	93	86	56	68	88	88	88	83	71	**76**
	XLNet	75	46	57	88	97	92	83	47	60	86	86	85	82	63	70
	Ensemble (RF)	71	57	65	89	96	91	78	48	61	86	85	86	80	68	73
	Ensemble (LR)	77	63	69	90	96	93	81	51	63	88	88	88	83	70	75

average scores of .89 and .81, respectively. The number of samples for the neutral class was significantly lower than for the other classes in the original train-set. For which, we can clearly observe the poor performance for that class using all the approaches in the table III. This controlled augmentation approach enables all the models to improve the neutral class's performances.

2) Stack Overflow dataset: In this dataset, we observe a different trend from the app review dataset when the basic as well as the controlled augmentation approaches, are applied. We do not observe any improvement in the basic and controlled augmentation approaches. This is because the imbalance percentage is higher than the Jira and the App reviews datasets. So, when data augmentation is applied multiple times on the dataset to oversample the undersampled classes, the quality of samples degrades. The best performing approach remains xlnet-base-cased in terms of macro-average F1 score. However, we do see a slight 1% improvement when basic augmentation is applied for the xlnet-base-cased PTT approach. The ensemble approaches do not improve the results of the PTT approaches as well.

3) Jira dataset: Compared to when trained on the original train-set, we observe that augmentation helped improve the results for all the PTT as well as ensemble approaches. We see the best improvement of 8% in weighted-average F1 score and 10% in macro-average F1 score is achieved by the xlnet-base-cased PTT model. We get the best results for the LR ensemble approach, which achieved a weighted-average and macro-average F1 score of .98. Similar to the App review dataset, we can observe significant improvement through augmentation for this dataset as well. Unlike the App review dataset, when the controlled augmentation approach is taken for the Jira dataset, we do not observe improvement across all PTT and ensemble approaches. The reason behind this, according to our observation, can be that after applying basic augmentation to the original dataset, the dataset size was increased. Still, the class imbalance was not significant compared to the app review dataset. For this reason, we observe better performance on the basic augmentation approach.

RQ2 Main Findings:
The augmentation approaches aid the performances of all the PTT approaches as well as the ensemble approaches in two out of three datasets in terms of weighted- and macro-average F1 scores with most significant improvements for the undersampled classes.

VI. Limitations

One of the limitations of our study is that we only used datasets that are publicly available from previous works. As a result, we are not able to ensure the quality of the manual annotations of the dataset. This limitation also extends to our

TABLE V: Results for the Jira dataset

Dataset	Class	Negative			Positive			Weighted-average			Macro-average		
	Model	P (%)	R (%)	F1 (%)	P (%)	R (%)	F1 (%)	P (%)	R (%)	F1 (%)	P (%)	R (%)	F1 (%)
Jira	Sentistrength-SE	99	72	84	62	99	76	88	81	81	81	86	80
	SentiCR	95	98	97	96	90	92	95	95	95	95	94	95
	BERT	94	97	95	93	85	89	93	93	93	93	91	92
	RoBERTa	98	96	97	91	95	93	96	95	95	94	95	95
	XLNet	89	100	94	99	72	83	92	91	90	94	86	88
	Ensemble (RF)	96	99	97	97	92	94	96	96	96	96	95	96
	Ensemble (LR)	96	98	97	96	91	93	96	96	96	96	94	95
Jira Basic Augmentation	BERT	98	99	98	97	96	96	98	98	**98**	97	97	97
	RoBERTa	97	99	98	98	94	96	97	97	97	98	96	97
	XLNet	99	99	99	97	97	97	98	98	**98**	98	98	**98**
	Ensemble (RF)	98	99	99	98	96	97	98	98	**98**	98	97	97
	Ensemble (LR)	98	100	99	99	96	97	98	98	**98**	99	98	**98**
Jira Controlled Augmentation	BERT	96	100	98	100	92	96	97	97	97	98	96	97
	RoBERTa	98	99	98	98	95	96	98	98	**98**	98	97	97
	XLNet	98	95	96	89	96	92	95	95	95	94	95	94
	Ensemble (RF)	99	99	99	97	98	97	98	98	98	98	98	**98**
	Ensemble (LR)	98	99	99	98	96	97	98	98	**98**	98	97	**98**

TABLE VI: Results for the Github dataset

Dataset	Class	Negative			Neutral			Positive			Weighted-average			Macro-average		
	Model	P (%)	R (%)	F1 (%)	P (%)	R (%)	F1 (%)	P (%)	R (%)	F1 (%)	P (%)	R (%)	F1 (%)	P (%)	R (%)	F1 (%)
Github	Sentistrength-SE	78	73	76	77	85	81	86	77	81	80	79	79	80	79	79
	SentiCR	89	67	76	78	92	84	87	85	86	83	83	82	84	81	82
	BERT	90	89	89	91	91	91	93	93	93	91	91	91	91	91	91
	RoBERTa	89	87	88	91	90	90	90	94	92	90	90	90	90	90	90
	XLNet	87	88	88	93	88	90	89	95	92	90	90	90	89	90	90
	Ensemble (RF)	91	90	90	92	91	91	92	94	93	92	92	**92**	92	92	**92**
	Ensemble (LR)	91	89	90	91	91	91	92	94	93	91	91	91	91	91	91

data augmentation technique. As mentioned in Chapter IV we augment existing data through adopting various approaches. So the quality of the augmented data relies on the quality of the original dataset.

Another potential limitation of our work is related to the random splitting of data in our experimental setup. We split the dataset randomly into a 70-30 ratio, where 70% of the dataset was used to train and the rest 30% was used to test. As the data is random in each split, in each run the results might vary. This can be addressed by using more rigorous techniques like k-fold which we plan on doing in the future.

VII. Conclusion and Future Works

Our work builds on the prior studies on sentiment analysis in software engineering perspective. In this study we provide a comparative study on performance of ensembled models and pre-trained transformer-based models. It also investigates the requirement and performance of data augmentation in fine-tuning ensembled and pre-trained models. The pre-training was done on four datasets - Stack Overflow posts , Mobile app review dataset (App), GitHub pull-request and commit comments (GitHub), Jira issue comments (Jira). Our experiments reveal that overall, ensemble models perform better than pre-trained transformer-based models in three out of the four original datasets, i.e., datasets without any augmentation, in terms of weighted- and macro-average F1 scores. Our results also demonstrate that the augmentation approaches aid the performances of all the pre-trained transformer model approaches along with the ensemble models in two out of three datasets in terms of weighted- and macro-average F1 scores.

Ensemble models have more potential to give better performance in sentiment analysis from a software engineering perspective. Thus we are interested in exploring more ensemble techniques in the future. It is evident that the existing datasets under recent study has noticable class imbalance issues. So, we also aim to explore other approaches of handling imbalance of dataset, specifically for software engineering texts. The quality and size of the dataset is also an important factor in the model performance, so we intend to work on generating a larger dataset and quality data annotation.

References

[1] W. Medhat, A. Hassan, and H. Korashy, "Sentiment analysis algorithms and applications: A survey," *Ain Shams engineering journal*, vol. 5, no. 4, pp. 1093–1113, 2014.

[2] M. R. Wrobel, "Towards the participant observation of emotions in software development teams," in *2016 Federated Conference on Computer Science and Information Systems (FedCSIS)*. IEEE, 2016, pp. 1545–1548.

[3] M. R. Islam and M. F. Zibran, "Towards understanding and exploiting developers' emotional variations in software engineering," in *2016 IEEE 14th International Conference on Software Engineering Research, Management and Applications (SERA)*. IEEE, 2016, pp. 185–192.

[4] S. F. Huq, A. Z. Sadiq, and K. Sakib, "Is developer sentiment related to software bugs: An exploratory study on github commits," in *2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2020, pp. 527–531.

[5] M. Ortu, B. Adams, G. Destefanis, P. Tourani, M. Marchesi, and R. Tonelli, "Are bullies more productive? empirical study of affectiveness vs. issue fixing time," in *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*. IEEE, 2015, pp. 303–313.

[6] M. R. Islam and M. F. Zibran, "Sentistrength-se: Exploiting domain specificity for improved sentiment analysis in software engineering text," *Journal of Systems and Software*, vol. 145, pp. 125–146, 2018.

[7] ——, “Deva: sensing emotions in the valence arousal space in software engineering text,” in *Proceedings of the 33rd annual ACM symposium on applied computing*, 2018, pp. 1536–1543.

[8] T. Ahmed, A. Bosu, A. Iqbal, and S. Rahimi, “Senticr: a customized sentiment analysis tool for code review interactions,” in *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2017, pp. 106–111.

[9] F. Calefato, F. Lanubile, F. Maiorano, and N. Novielli, “Sentiment polarity detection for software development,” *Empirical Software Engineering*, vol. 23, no. 3, pp. 1352–1382, 2018.

[10] T. Zhang, B. Xu, F. Thung, S. A. Haryono, D. Lo, and L. Jiang, “Sentiment analysis for software engineering: How far can pre-trained transformer models go?” in *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2020, pp. 70–80.

[11] S. Mishra and A. Sharma, “Crawling wikipedia pages to train word embeddings model for software engineering domain,” in *14th Innovations in Software Engineering Conference (formerly known as India Software Engineering Conference)*, 2021, pp. 1–5.

[12] J. Wei and K. Zou, “Eda: Easy data augmentation techniques for boosting performance on text classification tasks,” *arXiv preprint arXiv:1901.11196*, 2019.

[13] V. Carofiglio, F. d. Rosis, and N. Novielli, “Cognitive emotion modeling in natural language communication,” in *Affective information processing*. Springer, 2009, pp. 23–44.

[14] J. A. Russell, “A circumplex model of affect.” *Journal of personality and social psychology*, vol. 39, no. 6, p. 1161, 1980.

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

[16] 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)*. IEEE, 2017, pp. 203–214.

[17] B. Lin, F. Zampetti, G. Bavota, M. Di Penta, M. Lanza, and R. Oliveto, “Sentiment analysis for software engineering: How far can we go?” in *Proceedings of the 40th international conference on software engineering*, 2018, pp. 94–104.

[18] H. Batra, N. S. Punn, S. K. Sonbhadra, and S. Agarwal, “Bert-based sentiment analysis: A software engineering perspective,” in *International Conference on Database and Expert Systems Applications*. Springer, 2021, pp. 138–148.

[19] M. Thelwall, K. Buckley, G. Paltoglou, D. Cai, and A. Kappas, “Sentiment strength detection in short informal text,” *Journal of the American society for information science and technology*, vol. 61, no. 12, pp. 2544–2558, 2010.

[20] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, “Recursive deep models for semantic compositionality over a sentiment treebank,” in *Proceedings of the 2013 conference on empirical methods in natural language processing*, 2013, pp. 1631–1642.

[21] M. Pennacchiotti and A.-M. Popescu, “Democrats, republicans and starbucks afficionados: user classification in twitter,” in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2011, pp. 430–438.

[22] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.

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

[24] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, “Xlnet: Generalized autoregressive pretraining for language understanding,” *Advances in neural information processing systems*, vol. 32, 2019.

[25] L. Villarroel, G. Bavota, B. Russo, R. Oliveto, and M. Di Penta, “Release planning of mobile apps based on user reviews,” in *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*. IEEE, 2016, pp. 14–24.

[26] D. Pletea, B. Vasilescu, and A. Serebrenik, “Security and emotion: sentiment analysis of security discussions on github,” in *Proceedings of the 11th working conference on mining software repositories*, 2014, pp. 348–351.

[27] N. Novielli, F. Calefato, D. Dongiovanni, D. Girardi, and F. Lanubile, “Can we use se-specific sentiment analysis tools in a cross-platform setting?” in *Proceedings of the 17th International Conference on Mining Software Repositories*, 2020, pp. 158–168.