Effectiveness of Data Augmentation and Ensembling Using Transformer-Based Models for Sentiment Analysis: Software Engineering Perspective

Zubair Rahman Tusar[1,2], Sadat Bin Sharfuddin[1,2], Muhtasim Abid[1,2], Md. Nazmul Haque[1,2][a]
and Md. Jubair Ibna Mostafa[1,2][b]

[1]*Software Engineering Lab (SELab), Islamic University of Technology (IUT), Gazipur, Bangladesh*

[2]*Department of Computer Science and Engineering, Islamic University of Technology (IUT), Gazipur, Bangladesh*

Keywords: Sentiment Analysis, Pre-Trained Transformer-Based Models, Ensembling, Data Augmentation.

Abstract: Sentiment analysis for software engineering has undergone numerous research to efficiently develop tools and approaches for Software Engineering (SE) artifacts. State-of-the-art tools achieved better performance using transformer-based models like BERT, and RoBERTa to classify sentiment polarity. However, existing tools overlooked the data imbalance problem and did not consider the efficiency of ensembling multiple pre-trained models on SE-specific datasets. To overcome those limitations, we used context-specific data augmentation using SE-specific vocabularies and ensembled multiple models to classify sentiment polarity. Using four gold-standard SE-specific datasets, we trained our ensembled models and evaluated their performances. Our approach achieved an improvement ranging from 1% to 26% on weighted average F1 scores and macro-average F1 scores. Our findings demonstrate that the ensemble models outperform the pre-trained models on the original datasets and that data augmentation further improves the performance of all the previous approaches.

1 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 (Medhat et al., 2014). 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 years, the software engineering community devotes a substantial amount of effort to conducting research on sentiment analysis. Since software development relies heavily on human efforts and interactions, it is more vulnerable to the practitioners' emotions. Accurate sentiment analysis can help to improve various aspects of the software engineering domain that are affected by sentiment and emotion. It has been used to investigate the role of emotions in IT projects and software development (Wrobel, 2016)(Islam and Zibran, 2016), finding relations between developers' sentiment and software bugs (Huq et al., 2020), etc.

Numerous techniques and tools available for sentiment analysis that are being used in software engineering. These techniques mainly employ unsupervised, supervised, and Transformer-based approaches. Firstly, the unsupervised approach is used by tools like SentiStrength-SE (Islam and Zibran, 2018b) and DEVA (Islam and Zibran, 2018a). The authors leveraged domain-specific keyword dictionaries to achieve good performance in SE-specific texts, as these contain differences in meaning which are valuable for determining sentiment. For example, words like bug, patch, etc. can provide one kind of sentiment when used in SE-specific texts and another kind of sentiment when used in a general context, and having such domain-specific keywords in their dictionary made these tools perform better in SE-specific texts. Secondly, the supervised learning approach is used by tools like SentiCR (Ahmed et al., 2017) and Senti4SD (Calefato et al., 2018) which achieved improvement over the previous unsupervised approaches as these learned from the training dataset instead of relying on manually defined word-wise sentiment polarities. Despite the improvements, the proliferation of

[a] https://orcid.org/0000-0002-5120-3030
[b] https://orcid.org/0000-0003-2096-0474

438

Tusar, Z., Sharfuddin, S., Abid, M., Haque, M. and Mostafa, M.
Effectiveness of Data Augmentation and Ensembling Using Transformer-Based Models for Sentiment Analysis: Software Engineering Perspective.
DOI: 10.5220/0012092500003538
In *Proceedings of the 18th International Conference on Software Technologies (ICSOFT 2023)*, pages 438-447
ISBN: 978-989-758-665-1; ISSN: 2184-2833

transformer-based approaches achieved better results in other natural language domains attracts many researchers to utilize pre-trained transformer-based models in the software engineering domain. Among these models, fine-tuned BERT, RoBERTa, and XLNet outperformed existing unsupervised and supervised tools (Zhang et al., 2020) on SE-specific artifacts. The pre-trained models achieved 6.5% to 35.6% improvement in terms of macro and micro-averaged F1-scores (Zhang et al., 2020).

Although the pre-trained transformer models performed better than the previous approaches, small dataset size, and class imbalance problems incur threats to those models' performances. Moreover, existing research did not consider the ensembling of multiple models over individual models to know whether they have any impact on performance or not. To solve the class imbalance and small dataset size problems, we conducted a study on the effectiveness of data augmentation. We further conducted ensembling of the pre-trained transformer-based models to determine the impact on performance.

We evaluated the proposed approach using four publicly available datasets with annotated sentiment polarities. We chose three pre-trained transformer-based models: BERT, RoBERTa, and XLNet. We applied data augmentation with SE-specific Word2Vec (Mishra and Sharma, 2021) and EDA (Easy Data Augmentation) (Wei and Zou, 2019). Along with the original dataset, we used two versions of augmented datasets. In the first version, we did not consider the class imbalance issue and augmented the datasets equally for all the classes. In the second version, which we referred to as controlled augmentation, we considered the class imbalance issue and applied augmentation in a way to balance the classes along with increasing the size of the datasets. Finally, for all datasets, we performed sentiment analysis by ensembling the chosen transformer-based models and analyzed the obtained results over contemporary approaches. Specifically, we investigated the following research questions.

- **RQ1:** Do the ensemble models outperform the pre-trained models on the original datasets?
- **RQ2:** Does data augmentation have any impact on 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.
- Comparative study of the effect of controlled augmentation and inclusion of SE-based data augmentation on existing small datasets

Structure of the Paper: Section 2 discusses the related works. Section 3 describes datasets in detail. The methodology adopted for our research is elaborated in section 4. In section 5, we evaluate, analyze, and discuss the results of our experiments and present the main findings. Limitations are mentioned in section 6. Finally, we conclude by outlining the future work in section 7.

2 RELATED WORKS

We discussed the approaches of SE-based sentiment analysis from four perspectives: unsupervised approaches, Supervised approaches for SE-based sentiment analysis, pre-trained models of NLP for sentiment analysis, and data augmentation and the ensembling of pre-trained models.

2.1 Unsupervised Approaches

Unsupervised approaches are used in Senti Strength and Senti Strength SE.

Senti Strength: It is a lexicon-based sentiment analysis technique, which uses dictionaries of both formal terms and informal texts (like - slang, and emoticons)(Thelwall et al., 2010). In the dictionary, every word was assigned a specific sentiment strength. Then, the tool categorized sentences into positive and negative emotions and determined the strength of the emotions based on dictionaries and linguistic analysis. However, sentistrength could not perform well on SE-specific data as the dictionary did not contain words specific to software engineering.

Senti Strength SE: To evaluate the performance of SentiStrength on software artifacts, 151 Jira issue comments were analyzed(Islam and Zibran, 2017), which achieved less accuracy. Investigating the less accuracy of Sentistrength, they found 12 reasons, for which domain-specific meanings of words were most prevalent. So they built a modified version of SentiStrength by adding a domain-specific dictionary(Islam and Zibran, 2017). New sentiment words and negations were added to the dictionary. It was the first sentiment analysis tool where SE-specific context was considered. The dictionary, however, had very limited domain-specific words.

2.2 Supervised Approaches

Supervised approaches are discussed using Stanford CoreNLP, SentiCR, and Senti4SD tools. Each of these tools used supervision to classify sentiment and/or polarity.
Stanford CoreNLP: Stanford CoreNLP(Socher et al., 2013) was introduced for single-sentence sentiment classification; polarity along with the sentiment value of a sentence was returned by the tool. It was trained with the Recursive Neural Tensor Network on the Stanford Sentiment Treebank.
SentiCR: It was developed specifically for code review comments (Ahmed et al., 2017). It classified code review comments into two classes - negative and non-negative. The supervised classifier used in sentiCR was GBT(Gradient Boosting Tree)(Pennacchiotti and Popescu, 2011), as it gave the highest precision, recall, and accuracy among the eight evaluated classifiers.
Senti4SD: It was the first supervised learning-based tool to generate feature vectors. Senti4SD(Calefato et al., 2018) used three features - SentiStrength lexicons, n-gram extracted keywords from the dataset, and word representations in a distributional semantic model(DSM) exclusively trained on StackOverflow data. Four prototype vectors namely positive polarity word vectors, negative polarity word vectors, neutral polarity word vectors, and the sum of positive, and negative word vectors were calculated to extract semantic features. Finally, a Support Vector Machine (SVM) was used to identify sentiment polarities.

Although supervised approaches performed superiorly to unsupervised approaches, they were under par to properly predict sentiment in cases where sentiment-heavy words were not included in their training dataset.

2.3 Pre-Trained Models

There are some popular transformer-based pre-trained models that are described here for sentiment analysis.
BERT: A deep learning model designed to learn contextual word representations from unlabeled texts(Devlin et al., 2018). It is 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 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.
RoBERTa: A robustly optimized 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 Next Sentence Prediction loss, and train with dynamic masking. When Liu et al(Liu et al., 2019) released it, it had achieved state-of-the-art results on several benchmarks surpassing BERT.
XLNet: Based on Transformer-XL, it used segment recurrence mechanism and relative encoding. To address the individual weakness of autoregressive language modeling and autoencoding, it combined their strengths (Yang et al., 2019). XLNet performed better than BERT, including sentiment analysis, especially for long texts.

Although the size of the training dataset plays a crucial role in fine-tuning these large models, we observed less than one thousand samples even in the gold-standard datasets used in the community.

2.4 Data Augmentation and Ensembling of Pre-Trained Models

Data augmentation(Batra et al., 2021) was used 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. However, the class imbalance was evident in the datasets used in these approaches. Sample size for different sentiment polarities, i.e., positive, negative, and neutral, differed by a large margin. As a result, this issue caused 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.(Wei and Zou, 2019). The authors evaluated this technique and found that the improvement rate of this technique increases inversely proportional to the size of the dataset.

Table 1: Distribution of the datasets over the classes.

Dataset	Total	Positive	Neutral	Negative
App Review	341	186	25	130
Stack Overflow	1,500	131	1,191	178
Jira	926	290	0	636
Github	7,122	2,013	3,022	2,087

Table 2: Data samples from the Datasets under study.

Dataset	Samples from dataset	Usage in natural lang.
Github	Seems like something is **leaking** memory :(	The plumber fixed the **leaking** pipe.
Stack Overflow	I am attempting to get my JUnit tests for an Android application running using **Ant**.	**Ants** are all over my food
Jira	safety and scope **beans** to current **thread**.	White **thread** should have been used here
App Reviews	android 4.2**jelly bean** can't install error 909. . i'll give you five stars if u fix it	**Jelly bean** is one my favorite pastime food while watching movies

3 DATASET

In our study, we used four gold-standard publicly available datasets with annotated sentiment polarity. The distribution of the datasets over the classes (positive, neutral, and negative) is shown in Table 1. Some samples from these datasets are highlighted in Table 2 that demonstrate how software engineering context can influence the meaning of different words.
Jira Issues. The original dataset (Ortu et al., 2015) had four labels of emotions: love, joy, anger, and sadness which Lin et al.(Lin et al., 2018) brought down to two labels by annotating sentences with love and joy with positive polarity and sentences with anger and sadness with negative polarity.
App Reviews. 3000 reviews were presented (Villarroel et al., 2016) from which 341 reviews were randomly selected by Lin et al.(Lin et al., 2018). The dataset has a 5% confidence interval and a 95% confidence level making it statistically significant.
Stack Overflow Posts. There are 1500 sentences in total. This dataset was gathered by Lin et al. (Lin et al., 2018) 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.
GitHub Data. 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.(Pletea et al., 2014) by Novielli et al. (Novielli et al., 2020) to obtain annotated text units.

4 METHODOLOGY

In this section, we describe the proposed methodology to detect the sentiment polarities. It consists of three components. At first, the implementation details of the adopted augmentation approaches are discussed (4.1). Secondly, we elaborate on the fine-tuning step (4.2). Finally, we describe the implementation of the ensemble approach (4.3). An overview of our methodology is shown in Figure 1. The following subsections describe these three components with further details.

4.1 Augmentation

Motivated by the work of (Wei and Zou, 2019), we adopted the augmentation technique to mitigate the problem of small dataset size and class imbalance over the samples. The technique used three ways to introduce new samples from the given samples.

- **Random Synonym Replacement:** It was used to replace random words in a given sample (sentence) with their synonyms. We applied two different techniques. One was to replace randomly selected words with the synonyms found in Natural Language Toolkit (NLTK) wordnet library(Miller, 1995). Another one is to replace SE-specific words using most similar words in SE context provided by a word2vec model trained specifically on software engineering text from the study conducted by Siba m et al. (Mishra and Sharma, 2021). Some of the augmented results are shown in Table 4.
- **Random Deletion:** We randomly deleted a word from the given sentence. In addition, if the given sample contained duplicated words, those words were also discarded.
- **Random Insertion:** Like the other two ways, we randomly insert random words in a given sample. To do this, at first, we randomly pick a word from the given sentence and then, find the synonym of that word. Finally, we inserted this synonym into a random position.

We performed two types of augmentation only on the training dataset to generate a new dataset from the given raw dataset.
Basic Augmentation: It was used to tackle the issue of the dataset being small in size. We randomly picked a sample from the dataset and used either of the aforementioned three techniques to generate a new sample. However, this did not solve the class imbalance issues of the datasets.

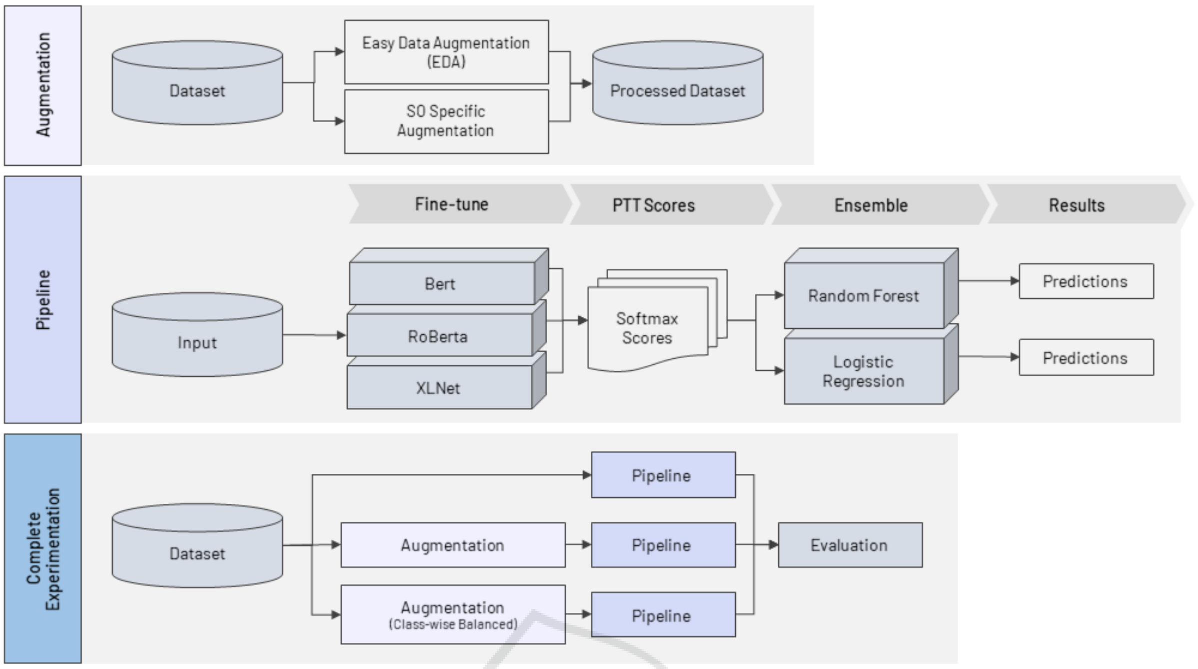
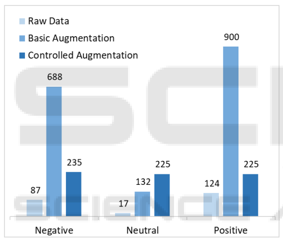
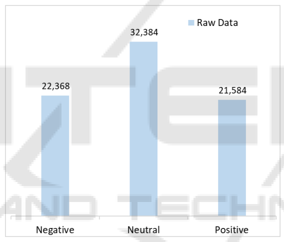


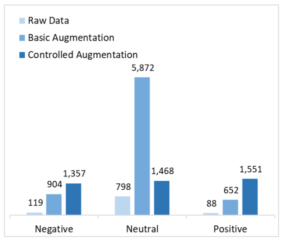
Figure 1: Overview of the proposed methodology.



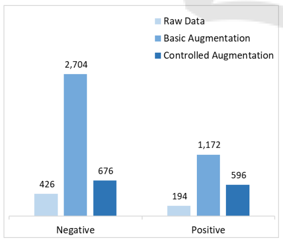
(a) Dataset: App Reviews.



(b) Dataset: Github.



(c) Dataset: Stack Overflow.



(d) Dataset: Jira.

Figure 2: Class frequency distributions for all datasets.

Controlled Augmentation: Class imbalance in many cases can cause bias problem in machine learning models. So, in this Controlled Augmentation, we applied augmentation techniques to balance the class-wise frequency distribution. For each dataset, we augmented text to ensure equal distribution in every class.

The distribution of samples over the classes after applying the three techniques and two types of augmentation is shown in Figure 2.

Table 3: Pre-trained transformer models and configurations.

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

4.2 Fine-Tuning Transformer-Based Models

The second step in our research methodology was to fine-tune the pre-trained transformer-based models for the downstream tasks. To do this, we split each dataset under our study into 70% for the train-set and 30% for the test-set. We fine-tuned three transformer-based models separately with the train sets split from the raw datasets as well as augmented datasets (generated using basic and controlled augmentation approaches). The models are BERT, RoBERTa, and XLNet. We refer to these models collectively as pre-trained transformers (PTT). The information related to train these models is shown in Table 3.

4.3 Ensemble

We proposed a 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.

Table 4: Data augmentation: SE-Specific Synonym Replacement utilizing word2vec (Mishra and Sharma, 2021).

Original	Augmented
A perfect timing. Just a second before my **commit** :)”	A perfect timing. Just the third during my **rollback** :)”
don’t pull. **master** is broken! will fix soon.”	don’t pull. **slave** is broken! will fix soon.”

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. At first, we considered every sample as a bag of words and encoded by the positional encoding that were represented in eq.1.

$$sample_i = [w_1, w_2, w_3, w_4 ..., w_n] \tag{1}$$

Then, we fed these n number of samples into m number of transformer based pre-trained models and took the softmax outputs of the last layer of these models, showed in Eq. 2 and 3. These outputs were concatenated and treated as feature vector (FV) which contained the contextual representation of the given input samples.

$$\begin{gathered} FV = \oplus_{i=1}^{m} \oplus_{j=1}^{n} softmax(models_i(sample_j)); \\ models = \{BERT, XLNet, RoBERTa\} \end{gathered} \tag{2}$$

$$\text{softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \tag{3}$$

Finally, we detected the sentiment polarities using two well known classifiers: Random Forest (RF) and Logistic Regression (LR) using Eq. 4.

$$\hat{Y}_i = classifiers_i(FV); classifiers = \{RF, LR\} \tag{4}$$

However, as three of the datasets (Stack Overflow, App review, and Github) under our study have three classes and LR is designed for binary classification, we adopted the one-vs-rest (OvR) method for multi-class classification.

5 EVALUATION AND DISCUSSION

In this section, we reported, compared and analyzed the performance of the pre-trained transformer-based models and the ensemble models on the four datasets described in Section 3. It was evident from **Figure 2** that three of the four datasets (App reviews, Stack Overflow, and Jira) had a small number of samples and a significant class imbalance issue. Thus, we conducted augmentation on these three datasets except for the Github (GH) dataset. For each dataset, we reported the performance of the approaches we used on the original version, the basic augmented version, and the controlled augmented version. For comparison, we also showed 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 highlighted the best performance in terms of the two main metrics (i.e., weighted-average F1 scores and macro-average F1 scores) in bold. We answered the research questions based on the experimental results as follows.

5.1 RQ1: Do the Ensemble Models Outperform the Pre-trained Models on the Original Datasets?

We answer RQ1 by analyzing the performance of ensemble models and PTTs only on the original version of the four datasets.

5.1.1 App-Review Dataset

The ensemble models outperform the PTT approaches in both weighted 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 (5). 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. Even though the dataset is small and the class distribution is highly imbalanced, the ensemble models still perform better compared to the other approaches.

5.1.2 Stack Overflow Dataset

The best-performing PTT approach is RoBERTa considering both weighted(89%) and macro-average(75%) F1 scores for this dataset, 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

Table 5: 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 6: 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

89% and 72%, respectively. Although RoBERTa outperforms the ensemble models, the ensemble models outperform the other two PTT approaches. More specifically, as illustrated in Table 6 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.

5.1.3 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. Table 7 clearly portrays 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.

5.1.4 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 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 (Table 8). 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.

Table 7: 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 8: 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

RQ1 Findings:
The ensemble models outperform the pre-trained transformer-based models by a significant margin for three out of the four datasets showing that ensemble models can perform better than individual transformer models despite having class imbalance. In the Stack Overflow dataset, RoBERTa, which is a PTT approach, performs better than the ensemble models.

5.2 RQ2: Does Data Augmentation Have any Impact on the Performances of the Models?

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

5.2.1 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 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, as shown in Table 5. 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, and the RF ensemble approach improved in the 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-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 5. This controlled augmentation approach enables all the models to improve the neutral class's performances.

5.2.2 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 the samples degrades. The best-performing approach remains XLNet-based PTT in terms of macro-average F1 score. However, we do see a slight 1% improvement when basic augmentation is applied for the XLNet-based PTT approach. The ensemble approaches do not improve the results of the PTT approaches as well.

5.2.3 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 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 the most significant improvements for the undersampled classes.

6 LIMITATIONS

One of the limitations was that we only used datasets that were publicly available from previous works. As a result, we were not able to ensure the quality of the manual annotations. This limitation also extended to our data augmentation technique. As an example, the text "Can't compile X64 under vs2010, returnng this error : 5>d:\mangos\mangos\src\game \spellmgr.cpp(1226): error C4716: 'DoSpellProcEvent::AddEntry' : must return a value" from the datasets under study may incline more towards a negative sentiment but was annotated as neutral. As mentioned in Section 4, we augmented existing data by adopting various approaches. So the quality of the augmented data relied 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.

7 CONCLUSION AND FUTURE WORKS

We provide a comparative study on the performance of ensembled models and pre-trained transformer-based models in terms of weighted and macro-average F1 scores. We also investigated the requirement and performance of data augmentation in fine-tuning ensembled and pre-trained models. The pre-training was done on four datasets. Experimental results revealed that ensemble models perform better than individual pre-trained models in three out of the four original datasets, i.e., datasets without any augmentation. Our results also demonstrated that the augmentation approaches aid the performances of all the pre-trained transformer models along with the ensemble ones in two out of three datasets.

Ensemble models have more potential to give better performance from a software engineering perspective. Thus, more ensemble techniques can be explored in the future. Besides, the existing datasets under recent study have noticeable class imbalance issues. So, we also aim to explore other approaches of handling imbalance of dataset. The quality and size of the dataset is also an important factor that can be furthered experimented on generating a larger dataset with quality data annotation.

REFERENCES

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

Batra, H., Punn, N. S., Sonbhadra, S. K., and Agarwal, S. (2021). Bert-based sentiment analysis: A software engineering perspective. In *International Conference on Database and Expert Systems Applications*, pages 138–148. Springer.

Calefato, F., Lanubile, F., Maiorano, F., and Novielli, N. (2018). Sentiment polarity detection for software development. *Empirical Software Engineering*, 23(3):1352–1382.

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

Huq, S. F., Sadiq, A. Z., and Sakib, K. (2020). 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)*, pages 527–531. IEEE.

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

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

Islam, M. R. and Zibran, M. F. (2018a). Deva: sensing emotions in the valence arousal space in software engineering text. In *Proceedings of the 33rd annual ACM symposium on applied computing*, pages 1536–1543.

Islam, M. R. and Zibran, M. F. (2018b). Sentistrength-se: Exploiting domain specificity for improved sentiment analysis in software engineering text. *Journal of Systems and Software*, 145:125–146.

Lin, B., Zampetti, F., Bavota, G., Di Penta, M., Lanza, M., and Oliveto, R. (2018). Sentiment analysis for software engineering: How far can we go? In *Proceedings of the 40th international conference on software engineering*, pages 94–104.

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

Medhat, W., Hassan, A., and Korashy, H. (2014). Sentiment analysis algorithms and applications: A survey. *Ain Shams engineering journal*, 5(4):1093–1113.

Miller, G. A. (1995). Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Mishra, S. and Sharma, A. (2021). 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)*, pages 1–5.

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

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

Pennacchiotti, M. and Popescu, A.-M. (2011). Democrats, republicans and starbucks afficionados: user classification in twitter. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 430–438.

Pletea, D., Vasilescu, B., and Serebrenik, A. (2014). Security and emotion: sentiment analysis of security discussions on github. In *Proceedings of the 11th working conference on mining software repositories*, pages 348–351.

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

Thelwall, M., Buckley, K., Paltoglou, G., Cai, D., and Kappas, A. (2010). Sentiment strength detection in short informal text. *Journal of the American society for information science and technology*, 61(12):2544–2558.

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

Wei, J. and Zou, K. (2019). Eda: Easy data augmentation techniques for boosting performance on text classification tasks. *arXiv preprint arXiv:1901.11196*.

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

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

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