IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 21 March 2024, accepted 28 April 2024, date of publication 6 May 2024, date of current version 15 May 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3397211

RESEARCH ARTICLE

Mining the Opinions of Software Developers for Improved Project Insights: Harnessing the Power of Transfer Learning

ZEESHAN ANWAR[1], HAMMAD AFZAL[1], TAHER AL-SHEHARI[2], MUNA AL-RAZGAN[3], TAHA ALFAKIH[4], AND RAHEEL NAWAZ[5]

[1]National University of Sciences and Technology, Islamabad 44000, Pakistan
[2]Department of Self-Development Skills-Computer Skills, Common First Year Deanship, King Saud University, Riyadh 11362, Saudi Arabia
[3]Department of Software Engineering, College of Computer and Information Sciences, King Saud University, Riyadh 11345, Saudi Arabia
[4]Department of Information Systems, College of Computer and Information Sciences, King Saud University, Riyadh 11543, Saudi Arabia
[5]Department of Digital Transformation, Staffordshire University, ST4 2DE Stoke-on-Trent, U.K.

Corresponding author: Hammad Afzal (hammad.afzal@mcs.edu.pk)

This work was supported by the Researchers Supporting Project, King Saud University, Riyadh, Saudi Arabia, under Grant RSP2024R206.

ABSTRACT Sentiment Analysis, a crucial tool for analyzing user opinions, has shown efficacy particularly when tailored to specific domains. While existing research predominantly focuses on training various classifiers for sentiment analysis within the software engineering (SE) domain, the outcomes often lack consistency when tested across different datasets. To address this gap, this paper proposes a novel approach utilizing transfer learning-based classifiers, fine-tuned and evaluated across diverse SE datasets. A comprehensive study is conducted, benchmarking machine learning and deep learning classifiers for SE sentiment analysis. Results indicate that transfer learning classifiers, namely GPT and BERT, outperform traditional approaches. Notably, the Bert large model achieves an F1-score of 0.89 on the Stack Overflow dataset, surpassing existing state-of-the-art tools. This research not only provides centralized insights but also paves the way for developing more accurate domain-specific sentiment analysis tools tailored for Software Engineering.

INDEX TERMS Domain-specific sentiment analysis, community dataset, deep neural networks, transformers.

I. INTRODUCTION

Sentiment analysis is a process of extracting the emotion of users from textual data. Sentiment Analysis techniques can be categorized into lexicon-based, machine-learning-based, and hybrid techniques [1]. Applications of sentiment analysis also vary from monitoring the reputation of a product to tracking voters' feelings and in financial markets [2]. Similarly, some applications of sentiment analysis in software engineering is evaluation of product features, requirements prioritization, and product rating [3], [4]. Existing research [5] shows that the accuracy and precision of domain-specific sentiment analysis are better than general-purpose tools as these tools' accuracy depends on the lexicon/dictionary. Furthermore, various machine learning techniques being the popular techniques for developing sentiment classifiers [6], [7], [8], [9], [10], [11], [12], [13], [14] for sentiment analysis were also studied, but the results of these techniques are quite diverse.

Sentiment analysis is valuable in software engineering because one can capture users' opinions about a particular application or feature [4]. The main sources of opinion data in software engineering are app reviews, bug reports, and community question-answering sites. Existing sentiment analysis tools give negative results when applied to the software domain [15]. Therefore, research is underway to develop domain-specific sentiment analysis tools for software engineering [16]. However, existing sentiment analysis tools developed for the software domain also have various limitations. Firstly, these tools typically demonstrate efficacy

The associate editor coordinating the review of this manuscript and approving it for publication was Francisco J. Garcia-Penalvo.

only within the confines of a particular dataset, limiting their generalizability and applicability across diverse contexts [17]. Moreover, their performance may be influenced by inherent biases present within the data, potentially skewing the analysis results [15]. Additionally, a notable challenge faced by existing tools is their difficulty in accurately computing negative and neutral sentiments, which can lead to imprecise or misleading analyses [18].

There is a need to overcome the above-stated limitations and develop a reliable sentiment analysis tool that works well on all software engineering datasets. Therefore, in this research, a large-scale study is conducted in which various classifiers are developed and fine-tuned by using different feature sets. The purpose of this effort is to develop a reliable sentiment analysis tool and evaluate the performance of various algorithms for domain specific sentiment analysis. These classifiers are trained on publicly available SentiSE dataset. It is observed that the performance of the transfer learning models is better than the machine learning and deep learning models for sentiment analysis. Therefore, transfer learning models are declared as the best models in this research and are used in further experiments of this research. The fine-tuned transfer learning models are validated using different experiments that were conducted using eight machine learning, and four deep learning models. In addition to this, different publicly available software engineering datasets are also used for validation to show the strengths of fine-tuned models on new datasets. The performance of the best models is also compared with the existing state-of-the-art sentiment analysis tools developed for the software domain. The Bert-Large model gives the best results as compared to state-of-the-art tools on the Stack Overflow and Jira datasets.

This research presents several significant contributions that aim to advance sentiment analysis within the software engineering domain:

- Unlike many existing studies that primarily focus on training sentiment analysis classifiers within specific domains, this research leverages transfer learning-based classifiers. This approach significantly enhances the accuracy and generalization of sentiment analysis models across various software engineering datasets. The F1-Score of Bert model is 0.89 that is higher than the various classifier based techniques. To the best of knowledge this is the first research that fine-tune GPT for domain specific sentiment analysis within software engineering.
- While previous works often report results on a single dataset on which it was trained, this study conducts a comprehensive benchmarking exercise. The proposed model is systematically evaluated on a wide range of machine learning and deep learning classifiers across multiple publicly available software engineering datasets. This approach cover the limitation of existing sentiment analysis tools.
- A newly developed, customized versions of GPT and BERT models specifically tailored for sentiment analysis in software engineering are introduced. Experiments demonstrate that these customized models outperform all state-of-the-art methods on the SE datasets, achieving higher accuracy and performance levels, thereby setting a new standard for sentiment analysis within the domain.
- By establishing a benchmark for sentiment analysis classifiers in software engineering, this research serves as a foundational reference point for future investigations. The performance of transfer learning classifiers is compared against existing state-of-the-art sentiment analysis tools in the SE domain. This comparison highlights the superiority of the proposed approach, particularly demonstrated by the significant improvement in F1-score achieved by the Bert large model on the Stack Overflow dataset.

The remaining paper is organized into the following sections. Section II covers the related work. Research methodology is given in Section III. A detail of experiments with different classifiers is given in Section IV. Best tools are validated with different datasets in V and comparing best classifiers with the existing state-of-the-art tools is made in Section VI. Finally, Section VII concludes this paper.

II. RELATED WORK

Feldman [2], explained the concepts of sentiment analysis. Various levels for sentiment analysis, i.e., document, phrase, aspect, and comparative, are explained with appropriate examples. The accuracy of sentiment analysis depends on the lexicon/dictionary. The application of sentiment analysis is to monitor the product's reputation and track voters' feelings and financial markets. The research issues, as highlighted by the researchers, are i) creating better models for compositional sentiment ii) Automatic entity resolution iii) Identifying text relevant to each entity in a document that discusses multiple entities iv) classifying sarcasm v) noisy text vi) current systems overlook the objective statements.

In addition, a range of machine learning techniques are explored for sentiment analysis, but the findings were varied. For instance, Vohra and Teraiya [7] conducted a comparative analysis of machine learning, lexicon-based, and hybrid sentiment analysis methods. They found that the hybrid approach yielded the highest accuracy results. Jagdale et al. [9] used Naive Bayes (NB) and Support Vector Machines (SVM) to classify Amazon product reviews into positive and negative sentiments and reported 98.17% accuracy with NB. Shirsat et al. [10] used NB and SVM for sentence-level sentiment analysis and reported an accuracy of 96.46% on the BBC news dataset. Guia et al [11] compared the four machine learning algorithms (NB, SVM, Decision Trees (DT), and Random Forest (RF)). The results of SVM are best with 89% accuracy, precision, and recall on the Amazon Mobile review dataset. Gaye and Wulamu [12] compared the performance of XGboost, RF, and SVM for sentiment analysis and found that the accuracy of XGboost

is higher than RF and SVM. In another paper, [13], SVM, DT, Bagging, Boosting, RF, and Maximum Entropy are applied for sentiment analysis on Amazon, Yelp, and IMDb datasets. The results of Maximum Entropy and Boosting are best on all datasets. In another paper [14], compared the traditional machine learning and deep learning techniques for sentiment analysis. NB, SVM, Long Short Term Memory (LSTM), and Convolutional Neural Network (CNN) were implemented with different features for sentiment analysis. The best accuracy of 0.735 and 0.706 was achieved with NB and CNN respectively.

Islam and Zibran [19], highlighted the problem of sentiment analysis tools and showed that sentiment analysis tools are created using specialized vocabulary. Therefore, these tools do not fit in each domain. The same is true for domain-specific sentiment analysis tools because, for example, SentiStrength-SE gives good results on the Jira dataset, and the accuracy of Senti4SD is high on the Stack Overflow dataset. Furthermore, these tools do not correctly compute negative sentiments.

Existing sentiment analysis tools are not suitable for the SE domain and produce negative results. To overcome this, the researchers developed a sentiment analysis technique for a software library recommender system. 40k sentences were taken from Stack Overflow and classified as positive, neutral, or negative. Sentiment analysis of phrases was performed using the Recursive Neural Network (RNN). The suggested method was tested on three different SE datasets, including comments on JIRA issues, evaluations of mobile apps, and Stack Overflow discussions. The method was compared with four state-of-the-art approaches: SentiStrength, NLTK, Stanford CoreNLP, and SentiStrength-SE. The observed unsatisfactory outcomes despite extensive training. Although the findings of each technique were comparable, none of them met the threshold for recommending them as APIs [15].

The general-purpose dataset used to train sentiment classifiers in the SE domain is biased toward categorizing neutral sentences as negative. The researchers created 4423 posts balanced Gold standard dataset that was manually created. Twenty million Stack Overflow posts were used to develop the distributed semantic model. The researchers built an SVM classifier that used lexicon-based, keyword-based, and semantic characteristics to get around this. 25x features based on information gain were chosen. SentiStrength, SentiStrength-SE, and Senti4SD were compared. Compared to SentiStrength, improvements in recall for the neutral class of 19% and precision for negatives of 25% are observed [18].

To determine which dictionary has a greater or lesser potential for the analysis of sentiments in Software Engineering literature, Islam and Zibran [16] compare the four distinct dictionaries (AFFIN, VERDA, Senti-Strength, MPQA). According to the results, the analysis of SE text is performed better by AFFIN than by MPQA, which performs poorly because of informal language.

A study by Islam and Zibran replicated the findings of the [16] study. The three most cutting-edge tools for sentiment analysis in the SE domain are evaluated. The researchers developed datasets for JIRA issue comments, Stack Overflow postings, Code review comments, and Java libraries. SentiStrength-SE, Senti4SD, and SentiCR sentiment analysis tools were evaluated by calculating the precision, recall, and F-score. Comparing the performance of SE-specific tools against SentiStrength at baseline. For the JIRA dataset, SentiStrength-SE had the highest accuracy, whereas Senti4SD had a high accuracy on the Stack Overflow dataset. Using SO data, Senti4SD's F-measure is 0.84. [17]

The conclusion is drawn from the existing research that sentiment analysis for the software domain needs to receive greater attention. Especially, the limitations to correctly identifying neutral and negative sentiments and showing good performance across diverse datasets. Therefore, BERT and GPT based sentiment analysis tool were developed and a large-scale study was conducted by implementing a variety of machine learning and deep learning classifiers for comparing the results. Machine learning and deep learning algorithms are compared with the proposed model because most of the researchers have used machine learning and deep learning algorithms to compute sentiment analysis. Transformers have demonstrated its merits in numerous NLP tasks and therefore, is chosen to compute the sentiment of SE text.

III. RESEARCH METHODOLOGY

Data collection, data pre-processing / cleaning, feature extraction, training the machine learning models, and making predictions using the trained models on new test data make up study methodology, as shown in Figure 1.

A. DATASETS

This research uses the publicly available SentiSE [20] datasets for training and testing the prediction models. These datasets consist of CSV files with headers including ID, Sentiment, and Text. The ID serves as a unique identifier, Sentiment denotes the label, and Text represents the content of each post, which can vary in length. The dataset comprises 13,507 posts gathered from several software engineering community websites.

Despite not employing traditional techniques like data resampling or class-weighted loss functions to address class imbalance, we leveraged the SentiSE dataset collected from various websites, ensuring inherent diversity and representativeness. The inclusion of data from multiple websites enhances the generalizability of the classifiers trained on the dataset. By exposing the model to diverse linguistic styles, topics, and community dynamics, it becomes more robust and adaptable to real-world scenarios beyond any single platform like Stack Overflow.

To partition the data, 70% was randomly selected for training, while the remaining 30% was reserved for testing. Additionally, 10% of the training data was further randomly allocated for validation. Table 1 outlines the attributes of the

TABLE 1. Training dataset.

Attribute	Total	Unique	Positive	Negative	Neutral
Posts	9453	-	2036	1770	5647
URLs	294	-	-	-	-
Emotions	465	-	347	118	-
User Mentions	259	-	-	-	-
Unigrams	196240	14078	-	-	-
Bigrams	186814	89133	-	-	-

TABLE 2. Test dataset.

Attribute	Total	Unique	Positive	Negative	Neutral
Posts	4052	-	881	772	2449
URLs	126	-	-	-	-
Emotions	198	-	156	42	-
User Mentions	109	-	-	-	-
Unigrams	84436	8429	-	-	-
Bigrams	80392	45086	-	-	-

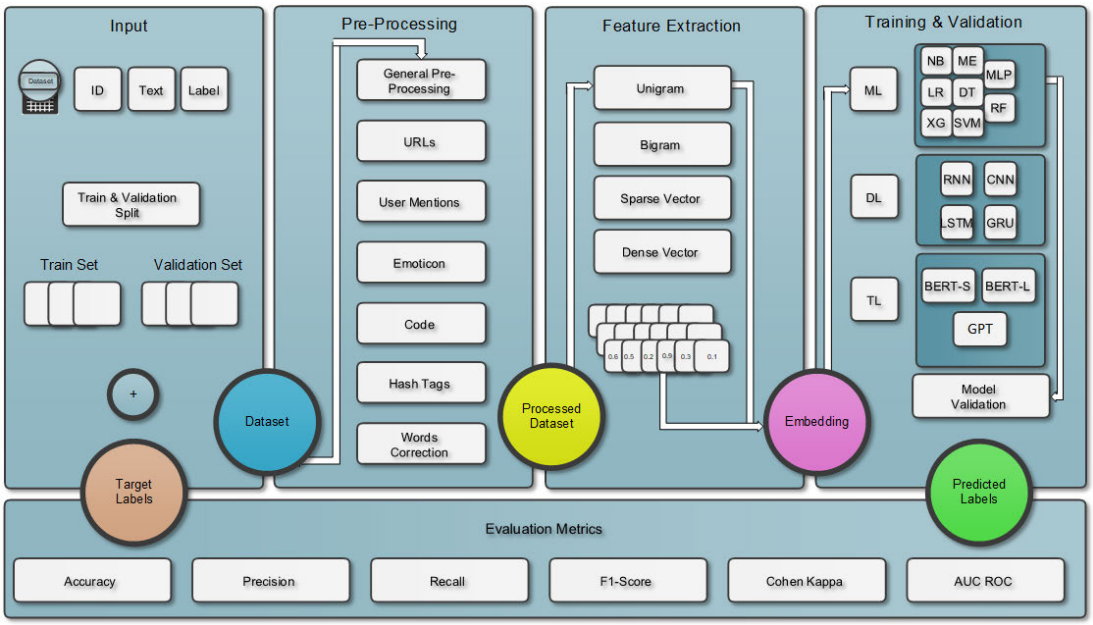


FIGURE 1. Proposed research methodology.

training dataset, which includes 2,036 positive posts, 1,770 negative posts, and 5,647 neutral posts. The test dataset, as shown in Table 2, encompasses 881 positive, 772 negative, and 2,449 neutral posts.

B. DATA PRE-PROCESSING

Due to people's casual attitudes, social media data is typically a noisy dataset. There are numerous emoticons, user mentions, codes, URLs, etc. in there. Data must be pre-processed in order to remove noise and transform it into a compressed and normalized form. The pre-processing [21] affects the accuracy of different machine learning classifiers. Hence, the data undergoes meticulous processing to improve the accuracy of the classifiers. Pre-processing steps can be divided into standard and specific pre-processing. The data undergoes standard preprocessing operations, including conversion to lowercase and the replacement of two or more consecutive dots (.) with spaces. Additionally, any trailing whitespace and quotation marks at the end of posts are removed and replaced with a single space. In addition to this, the special pre-processing steps are as follows:

- URL: Users frequently include URLs in postings, but since URLs are not necessary for text sentiment analysis, a regular expression (www.[\S]+)|(https?://[\S]+) is used to search for URLs and replace them with the term URL.
- User Mention: Users frequently mention another user in their reviews. @handle is typically used for user mentions. Similar to URLs, user mentions are not helpful for sentiment analysis, thus user mentions are replaced with the phrase USER_MENTION using regular expression (@[\S]+).

- Emoticon: Users can communicate their emotions by using emojis. The sentiment analysis process benefits from these feelings. Therefore, a list of emoticons is compiled, and replaced in the posts that matched them. Positive emoticons are changed to EMO_POS and negative ones to EMO_NEG as, respectively.
- Code: Code also doesn't help with sentiment analysis, therefore it is removed from user posts.
- Hashtag: Although hashtags are not common on SE community sites, certain users may use them to make a topic trend. As a result, the regular expression #(\S+) is used to delete the hash (#) sign from the hashtag.
- Individual Words: Each word is processed individually after using the pre-processing procedures mentioned above. The word is considered to be valid if it begins with the alphabet; else, it was eliminated. The words were punctuated with ['"?!,.():;] transformed repeating characters that were two or more into two characters. For instance, ''sooooo'' becomes ''soo''.

C. FEATURE EXTRACTION

Features extraction step is followed for the machine learning and deep learning models only. Depending on the classification method, unigram, and bigram features are extracted from the dataset and converted into feature vectors, i.e. sparse vector representation or dense vector representation. The training dataset contains 196240 unigrams. Because most unigrams with low-frequency distributions generate noise, the frequency distribution is used to select the top N unigrams for analysis. The value of N is chosen as 15000 and 90000 for sparse vector and dense vector representation respectively. The training dataset contains 186814 bigrams. Frequency distribution, like unigram, is used to select the top N bigrams and remove noise words from bigram. As a result, a total of 10000 bigrams are chosen.

D. TRAINING CLASSIFIERS

Several classifiers were trained in this research. Transfer learning classifiers (BERT and GPT) were trained to developed a novel tool for sentiment analysis of software engineering domain. Machine learning and deep learning were trained in this research to compare the performance of these classification models with the proposed transfer learning classifier. 70% data is used for training and 30% data is used for testing. 10% of training data is used for validation to avoid over-fitting. Sparse vector representation is used for machine learning classifiers and dense vector representation for deep learning classifiers.

1) TRAINING TRANSFER LEARNING CLASSIFIERS

Transformer is a method for pre-training language models. Several transformers like BERT [22] and Generative Pre-Training Transformer (GPT) [23] are available freely. These pre-trained models can be used to extract language features from data, or they can be fine-tuned for tasks such as classification, entity recognition, question answering, and so on [24]. In this work, domain-specific sentiment classifiers are fine-tuned and trained with BERT variants and GPT. This is accomplished by employing two BERT pre-trained models and a GPT model. Fine-tuning is accomplished by inserting an untrained neuron layer at the end of the pre-trained model and then retraining the model for the classification task. Training process of transformers employed in this research is given in Figure 2.

2) TRAINING MACHINE LEARNING CLASSIFIERS

Eight machine learning classifiers are implemented by using the scikit-learn library [25]. Four experiments are performed by training each classifier by using unigram, bigram, frequency, and presence feature types. In total 32 x different iterations of variants of eight machine learning classifiers were developed. Input to each model is processed data and output of these model is sentiment class for the particular instance of data. The implementation detail of each machine learning model is given in Validation Section V. These machine learning classifiers were implemented for validation of the proposed model with machine learning algorithms because in the existing literature various machine learning algorithms are utilized to train sentiment analysis classifiers.

3) TRAINING DEEP LEARNING CLASSIFIERS

Two distinct types of deep learning classifiers, namely the Convolutional Neural Network (CNN) [26] and Recurrent Neural Network (RNN) [27], are employed in this research for the validation of the proposed model with deep learning models. These classifiers are instantiated utilizing the Keras framework with a TensorFlow backend. The training of these models involves the use of dense vector representation, utilizing the top 90,000 words from the training dataset. The vocabulary assigns each word a unique integer index within the range of 1 to 90,000 while reserving index 0 for special padding words. Each word in the vocabulary is represented by a 200-dimensional vector.

The initial layer of the deep learning architecture comprises an embedding layer, initialized with weights randomly drawn from $N(0, 0.01)$. The rows within the embedding matrix, structured with dimensions $(v + 1) \times d$ (where v signifies the vocabulary size and d represents the word vector dimension), are populated with GloVe vectors sourced from the work of Pennington et al. [28] aligned with the words that match GloVe's word vectors. To accommodate variable post lengths, padding with zeros is applied at the end of the posts, standardizing all posts to a uniform length, denoted as max_length. The model's loss function is categorical cross-entropy, and the Adam optimizer is employed for weight updates. After experimenting with different training epochs, adjustments are made, setting the optimal epoch to 8 to mitigate over-fitting issues observed beyond this threshold.

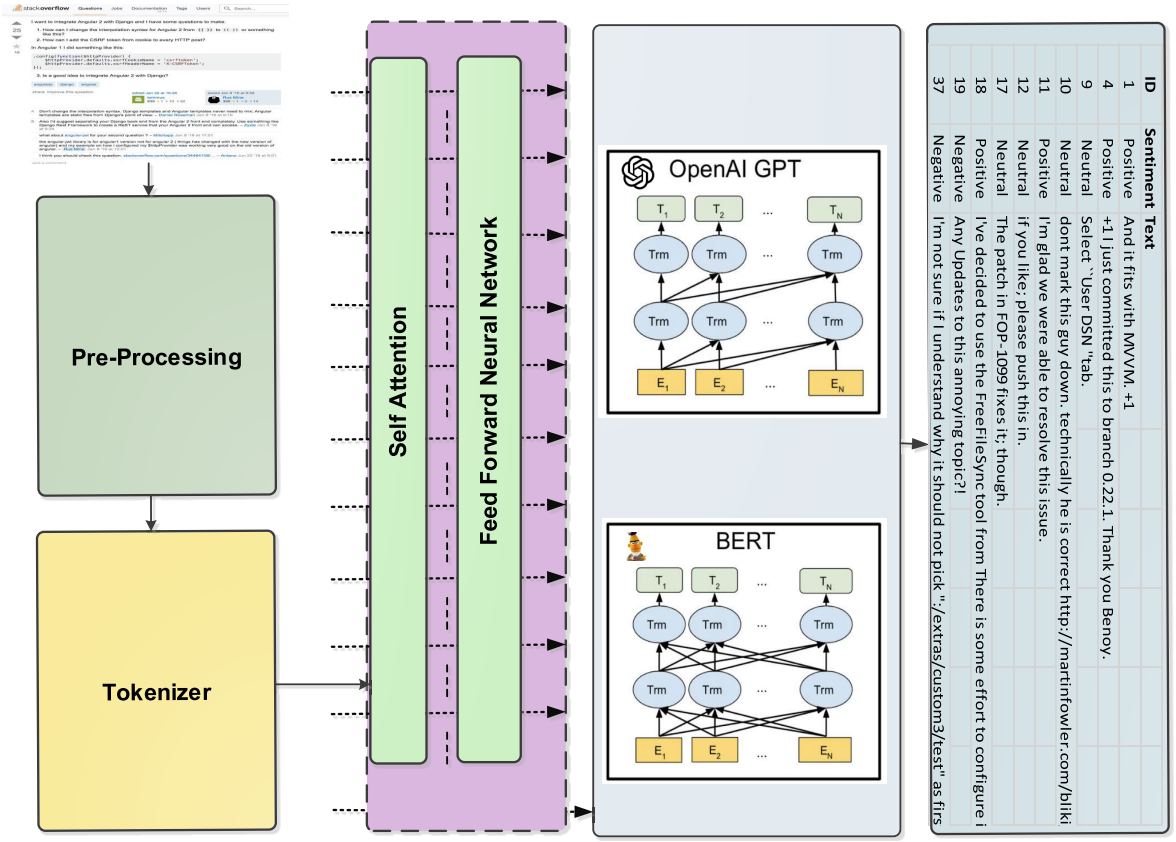


FIGURE 2. Architecture of BERT and GPT for sentiment analysis.

E. EVALUATION METRICS

Prediction models' accuracy, precision, recall, F1-Score, Cohen Kappa, and AUC ROC were determined to validate them. The best value for these metrics is 1. This means that the model is entirely correct.

IV. EXPERIMENTS

Python is used as a programming language for implementation in all of the experiments. Scikit-learn, Keras, Tensorflow, and NLTK library are used to implement different classifiers. 10% of training data is used for validation to avoid overfitting i.e. 8508 posts are used for training and 945 posts for validation. 4052 posts are used for testing. For each method, accuracy, precision, recall, F1-score, Cohen Kappa, and ROC AUC are calculated to evaluate the results. The details of experiments and results on validation and test dataset are given below:

A. FINE-TUNING BERT MODELS

The Hugging Face library [29], that is the PyTorch interface for working with BERT is used in the BERT experiments. For classification, a single layer of neurons is added to the top of BERT in this model. When domain-specific data is fed into the BERT and classification layer, the entire BERT and classification layer is trained for a specific task.

TABLE 3. BERT and GPT control parameters for fine-tuning.

Parameter	Value
Model	BERT_Base, BERT_Large, GPT
Batch Size	32
Epsilon parameter	1e-8
Max_Length	256
Epochs	4
Learning Rate	2e-5
Optimizer	AdamW

In the experiment, a BERT tokenizer is used to tokenize the data, added SEP and CLS tokens at the beginning and end of sentences, and padded the tokens with special [PAD] tokens to a fixed length. Experiment with various lengths, including 64, 128, 256, and 512 for the selection of optimal sentence length were conducted. An Attention mask is also used to distinguish between real and padded tokens. The BERT self-attention mechanism does not interpret sentences using padded tokens differentiated in attention mask. Table 3 contains the BERT fine-tuning and training hyperparameters. For domain-specific sentiment analysis, both the BERT Base Uncased and the BERT Large Uncased models are fine-tuned. These models are trained on 256 sentence lengths. BERT-Large gives the highest accuracy in this experiment.

TABLE 4. Results of different classifiers on validation & test dataset.

Algorithm	Val. Accuracy	Test Accuracy	Precision	Recall	F1	C Kappa	AUC
Bert-Base	0.850	0.846	0.823	0.818	0.820	0.722	0.861
Bert-Large	0.840	0.845	0.816	0.827	0.821	0.723	0.867
GPT	0.861	0.814	0.834	0.834	0.819	0.820	0.723

B. FINE TUNING GPT MODEL

The Hugging Face library is used in the GPT experiment, that is the PyTorch interface for working with transformers. For classification, a single layer of neurons is added to the top of model. When domain-specific data is fed into the transformers and classification layer, the entire model and classification layer is trained for a specific task. Table 3 contains the BERT and GPT fine-tuning and training hyperparameters.

The Table 4 presents the performance metrics of three different classifiers, namely Bert-Base, Bert-Large, and GPT, on both the validation and test datasets. The validation accuracy measures how well each classifier predicts the sentiment labels on a separate validation dataset, while the test accuracy evaluates their performance on an unseen test dataset.

Bert-Base and Bert-Large demonstrate similar validation and test accuracy rates, with Bert-Base achieving a validation accuracy of 85.0% and a test accuracy of 84.6%, and Bert-Large achieving 84.0% and 84.5%, respectively. On the other hand, GPT exhibits a slightly higher validation accuracy of 86.1%, but its performance decreases on the test dataset to 81.4%.

In terms of precision, recall, and F1-score, Bert-Base and Bert-Large exhibit consistent performance, with precision ranging from 81.6% to 82.3%, recall from 81.8% to 82.7%, and F1-score hovering around 82.0%. GPT, while showing slightly higher precision at 83.4%, matches its recall but has a slightly lower F1-score of 81.9%.

Cohen's Kappa coefficient, which measures agreement beyond chance, indicates moderate agreement for all three classifiers, with values ranging from 0.722 to 0.723. Additionally, the Area Under the Curve (AUC) values, reflecting the classifiers' discriminative ability, are relatively high, ranging from 0.861 to 0.867.

Overall, the results suggest that Bert-Base and Bert-Large classifiers perform consistently well across various metrics on both validation and test datasets. While GPT exhibits higher validation accuracy, its performance on the test dataset is slightly inferior. However, the F1-score of Bert-Large model is greater than Bert-Base and GPT. From this one can conclude that Bert-Large model is better for the domain specific sentiment analysis therefore, in the subsequent sections Bert-Large model is selected for further analysis.

V. BENCHMARKING AND VALIDATION

To benchmark the proposed domain specific sentiment analysis model various machine learning and deep learning algorithms with different combination of features were implemented. The results of the each algorithm was compared with the proposed model. Whereas, for the validation of the proposed model three different software engineering dataset were used. This method shows the performance of the proposed model on different dataset.

A. BENCHMARKING WITH MACHINE LEARNING CLASSIFIERS

Implementation detail, training parameters and results of machine learning algorithms implemented for domain specific sentiment analysis are given in this section. The results show that due to low performance most of the machine learning algorithms are not suitable for domain specific sentiment analysis.

1) NAÏVE BAYES

Naïve Bayes [30] is a simple and powerful classifier that can be used for multiple purposes. This research used it for text classification. The scikit-learn library [25] is utilized to implement MultinomialNB Naïve Bayes (NB) with Laplace smoothing. The default value of the smoothing parameter α is used in the experiments. In this classifier, the class $\hat{c}$ is assigned to text/community post t, as given in Eq (1).

$$\hat{c} = \underset{c}{argmax} P(c|t) \tag{1}$$

$$P(c|t) \propto P(c) \prod_{i=1}^{n} P(f_i|c) \tag{2}$$

In the Eq (2), maximum likelihood estimates is used to obtain the values of $P(c)$ and $P(f_i|c)$. f_i is the i^{th} feature out of total n features.

Different experiments with NB using unigram and, bigram features with a combination of presence and frequency feature types were conducted. The best results are achieved by using the presence features types because NB works better on integers. In Table 5 only the best results are reported. The accuracy using unigram on the validation set is 0.741 and on the test set is 0.756 whereas, the accuracy using bigram on the validation set is 0.751 and on the test set is 0.760.

2) LOGISTIC REGRESSION

Logistic Regression (LR) [31] is a statistical classification algorithm. It can work with binary data i.e. binary classification and ordinal data for multi-class classification. LR is implemented in Keras. Sequential model, the activation function is sigmoid, categorical_crossentropy loss, and Adam's optimizer are parameters of the LR model. For three classes

TABLE 5. Benchmarking of different classifiers on test dataset.

Algorithm	Accuracy	Precision	Recall	F1	C Kappa	AUC
NB (U)	0.756	0.749	0.654	0.682	0.523	0.740
NB (B)	0.760	0.730	0.691	0.705	0.552	0.766
LR (U)	0.537	0.394	0.376	0.374	0.071	0.532
LR (B)	0.533	0.389	0.374	0.372	0.067	0.530
MaxEnt (U)	0.682	0.734	0.489	0.509	0.294	0.617
MaxEnt (B)	0.242	0.503	0.360	0.176	0.024	0.518
DT (U)	0.787	0.780	0.687	0.718	0.585	0.767
DT (B)	0.784	0.768	0.683	0.710	0.581	0.765
RF (U)	0.762	0.760	0.645	0.682	0.525	0.735
RF (B)	0.776	0.768	0.672	0.705	0.561	0.755
XGBoost (U)	0.820	0.809	0.755	0.777	0.662	0.816
XGBoost (B)	0.822	0.811	0.756	0.779	0.664	0.816
SVM (U)	0.775	0.794	0.662	0.702	0.550	0.746
SVM (B)	0.774	0.806	0.653	0.696	0.544	0.740
MLP (U)	0.582	0.395	0.347	0.308	0.023	0.509
MLP (B)	0.572	0.387	0.351	0.323	0.031	0.513
RNN	0.775	0.740	0.702	0.716	0.579	0.776
CNN	0.824	0.800	0.783	0.789	0.680	0.836
Bert-Large	0.845	0.816	0.827	0.821	0.723	0.867

equation of LR can be represented as given in Eq(3).

$$Pr(Y_i = 1|X_i) = \frac{exp(\beta_0 + \beta_1 X_i + \beta_2 X_2 + \beta_3 X_3)}{1 + exp(\beta_0 + \beta_1 X_i + \beta_2 X_2 + \beta_3 X_3)} \quad (3)$$

The comparative analysis between frequency and presence features reveals a negligible difference in the obtained results. Detailed outcomes from the Logistic Regression (LR) model using both unigram and bigram features have been tabulated in Table 5. The test accuracy, marked at 0.573, indicates a low performance.

3) MAXIMUM ENTROPY

Maximum Entropy Classifier as introduced by [32] and based on the Principle of Maximum Entropy, operates by selecting the most uniformly probabilistic model that maximizes entropy under specified constraints. Unlike models assuming conditional independence among features, this classifier doesn't rely on such assumptions, thus negating concerns related to feature overlap. In binary classification tasks, the Maximum Entropy method aligns with logistic regression, facilitating the determination of distributions across classes. The Maximum Entropy model is typically denoted by the equation referenced as Eq (4).

$$P_{ME}(c|d, \lambda) = \frac{exp[\sum_i \lambda_i f_i(c, d)]}{\sum_{\hat{c}} exp[\sum_i \lambda_i f_i(c, d)]} \quad (4)$$

MaxentClassifier from NLTK library [33] is used to perform sentiment analysis on given data. MaxentClassifier is tested with unigram and bigram features. The results of classification are shown in Table 5. The best validation accuracy of 0.703 and 0.682 test accuracy by using unigram features is achieved.

4) DECISION TREE

Decision trees [34] are supervised classification algorithms used in labeled data to construct a decision tree structure. In this methodology, each node within the decision tree corresponds to a test on the dataset's attribute, and the subsequent branches represent the outcomes. Ultimately, the leaf nodes define the final class of the data. The selection process at each node involves determining the optimal test condition or decision. In the implementation of Decision Trees (DT), the Scikit-learn DecisionTreeClassifier is used. The evaluation of splits at individual nodes is performed using the GINI factor, aiding in the selection of the most suitable split. In the decision-making process for a given node t, the *GINI* factor, as represented in Eq (5), guides the selection of the optimal split. The calculation of the $GINI_{split}$ is outlined in Eq (6), ultimately leading to the selection of the split that minimizes the *GINI* factor.

$$GINI(t) = 1 - \sum_j [p(j|t)]^2 \quad (5)$$

where $p(j|t)$ is the relative frequency of sentiment class j at node t, and

$$GINI_{split} = \sum_k^{i=1} \frac{ni}{n} GINI(i) \quad (6)$$

In the Eq (6) ni = number of records at child i, n = number of records at node p which indicates the quality of the split. Results of the Decision tree on unigram and bigram features are given in Table 5. The best accuracy of 0.787 on validation data and 0.784 on test data is obtained by using bigram features.

5) RANDOM FOREST

Random Forest (RF) [35] is used for classification and regression. It is an ensemble learning algorithm. Random Forest generates many decision trees and classifies them based on the aggregate of these decision trees. The working of random forest can be explained as, for a set of posts represented as $x1, x2, \ldots xn$ and sentiment label $y1, y2, \ldots yn$ associated with these posts a random sample (X_b, Y_b)

(where b ranges from $1 \ldots B$) is repeatedly selected by bagging with replacement. Each classification tree is trained by using a random sample. Finally, a majority vote is taken on predictions of these B trees. Random Forest (RF) is also implemented by using Scikit-learn RandomForestClassifier. Frequency and presence features are used and experiment with 10 estimators. The best results are achieved by using presence features. The validation and test accuracy of RF by using unigram are 0.762 whereas, the validation accuracy of bigram is 0.748 and test accuracy is 0.776. Complete results of RF are shown in Table 5.

6) XGBOOST

XGBoost [36] operates as a gradient-boosting algorithm. The resultant prediction model generated by XGBoost comprises an ensemble of weak predictive decision trees. The ensemble, represented by K models, is described in Equation (7).

$$\hat{y}_i = \sum_{k}^{k=1} f_k(x_i), f_k \epsilon F \tag{7}$$

In the about Equation, $\hat{y}_i$ is the final output x_i is input and F is the space of trees. The goal is to minimize the following loss function as given in Eq (8).

$$L(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum \Omega(f_k) \tag{8}$$

where $\Omega(f_k)$ can be computed by Eq (9).

$$\Omega(f_k) = \gamma T + \frac{1}{2}\lambda \|w\|^2 \tag{9}$$

XGBoost classifier is also implemented for unigram and bigram features. So far the best results are achieved with the XGBoost classifier. Maximum tree depth to 25 and tune number of the estimator to 400 is set as this configuration gives the best results. The results of XGBoost are also given in Table 5. A maximum accuracy of 0.834 on validation data and 0.820 on test data by using unigram features is the output of XGBoost.

7) SUPPORT VECTOR MACHINE

Support Vector Machines (SVM) [37], were initially designed for binary classification and is a non-probabilistic binary linear classifier. SVM is also extended for multi-class classification [38]. The goal is to find the maximum-margin hyperplane (as given in Eq (10)) for a training set (x_i, y_i) that divides the points with $y_i = 1$ and $y_i = -1$.

$$w.x\ b = 0 \tag{10}$$

The equation to maximize the margin λ is given in Eq (11).

$$\underset{\omega,\gamma}{max}\gamma, s.t. \forall i, \gamma \leq y_i(w.x_i + b) \tag{11}$$

SVM is implemented by using the Scikit-learn library. The penalty parameter of the error term (C term) is set to 0.1 for all experiments. C term influences the misclassification of the objective function. SVM is also tested for unigram and bigram features. The results of unigram and bigram are comparable. The best accuracy on the validation dataset by using unigram features is 0.781 and the accuracy on the test dataset is 0.775.

8) MULTI-LAYER PERCEPTRON

Multilayer Perceptron (MLP) [39] belongs to the category of feed-forward neural networks featuring a minimum of three layers of interconnected neurons. Each neuron within the network applies a non-linear activation function and undergoes learning through a back-propagation algorithm. The computation of the MLP's output can be derived utilizing Equation (12). Known for its effectiveness in tackling intricate classification tasks, the MLP excels in learning non-linear models, notably beneficial in applications like sentiment analysis.

$$y = \varphi(\sum_{i=1}^{n} w_i xi + b) = \varphi(W^T X + b) \tag{12}$$

where w is vector of weights, x is input vector, b is bias and φ is activation function.

MLP is implemented using Keras with the TensorFlow backend. A 1-hidden layer neural network with 500 hidden units is used in the experiment. The model generates three outputs that pass through the sigmoid. Categorical crossentropy loss and Adam optimizer are used. The model is trained to 20 epochs. The results of MLP are comparable with logistic regression and are given in Table 5.

The Table 5 presents the benchmarking results of various classifiers on the test dataset. Across the classifiers, there is a notable variation in performance metrics. Naive Bayes (NB) classifiers, both unigram (U) and bigram (B) versions, show moderate accuracy, with NB (B) slightly outperforming NB (U). Logistic Regression (LR) models exhibit relatively lower performance, with both unigram and bigram versions showing similar accuracy rates. MaxEnt (Maximum Entropy) models perform reasonably well in unigram settings but demonstrate significantly degraded performance with bigram features. Decision Tree (DT) and Random Forest (RF) classifiers showcase solid accuracy, precision, recall, and F1-score values, with both unigram and bigram versions performing consistently well. Notably, XGBoost models achieve the highest accuracy among all classifiers, with both unigram and bigram versions exhibiting strong performance across all metrics. Support Vector Machine (SVM) classifiers and Multilayer Perceptron (MLP) neural networks display mixed performance, with SVM achieving moderate accuracy but lower precision and recall, and MLP exhibiting comparatively lower accuracy and precision.

Among all classifiers, Bert-Large, based on the BERT language model, stands out with the highest accuracy of 84.5% and solid precision, recall, and F1-score values, indicating its effectiveness in sentiment analysis tasks. However, it's noteworthy that XGBoost classifiers achieve comparable accuracy to Bert-Large while offering slightly

higher precision and F1-score values. Overall, the results suggest that ensemble methods like XGBoost and deep learning architectures like Bert-Large are particularly well-suited for sentiment analysis tasks on the given dataset, showcasing their potential for real-world applications in natural language processing tasks.

B. BENCHMARKING WITH DEEP LEARNING CLASSIFIERS

Two variants of RNN i.e LSTM & GRU and CNN deep learning models are developed for benchmarking of the proposed Bert-Large model. The implementation detail of these models is given as follow.

1) CONVOLUTIONAL NEURAL NETWORKS

Convolutional Neural Networks (CNNs) [26] represent a unique class of neural networks characterized by layers referred to as convolution layers. These layers possess the ability to discern spatial data patterns, and CNNs typically require minimal pre-processing of data. Each convolution layer is equipped with multiple filters or kernels that it learns to use in extracting specific features from the input data. The convolution operation is executed by sliding a 2D window-like kernel over the input data. For the experimental setup involving CNN, temporal convolution is employed, making it well-suited for analyzing sequential data.

The implementation of the Convolutional Neural Network (CNN) [40] is carried out using Keras with the TensorFlow backend. Various CNN architectures, denoted as 1-Conv-NN, 2-Conv-NN, 3-Conv-NN, and 4-Conv-NN, are explored in the experimentation. Among these, the most favorable results are obtained with the 4-Conv-NN architecture. A detailed overview of the 4-Conv-NN model's structure is provided in Figure 3.

The accuracy of the 4-layer CNN model on the validation set is 0.828 and on the test set is 0.824. Other metrics of the CNN model are given in Table 5.

2) RECURRENT NEURAL NETWORKS

Recurrent Neural Networks (RNN) constitutes a network of interconnected neuron-like nodes, with each node establishing directed connections to every other node. The fundamental RNN structure involves a hidden state, denoted as s_t, serving as the network's memory, capable of assimilating contextual information crucial for NLP classification tasks. At each time step, the output is computed based on the current input x_t and the memory s_t. The distinctive aspect of an RNN lies in its hidden state, enabling the capture of sequential dependencies within information.

In experimental settings, a specialized form of RNN known as the Long Short-Term Memory (LSTM) network is employed. LSTMs possess the ability to retain information over long sequences, which is beneficial in neural network tasks. The experiments involve the utilization of a neural network with two LSTM layers having output shapes of 128 and 64. The model is trained using categorical cross-entropy as the loss function, and the weights are updated using the Adam optimizer. With the integration of LSTM, the model achieves the highest accuracy of 0.791 on the validation dataset and 0.775 on the test dataset.

The Table 5 presents benchmarking results of different classifiers on a test dataset, showcasing their performance across various metrics. Among the classifiers evaluated, Recurrent Neural Network (RNN) achieves a solid accuracy of 0.775 with corresponding precision, recall, and F1-score values of 0.740, 0.702, and 0.716, respectively. Convolutional Neural Network (CNN) demonstrates superior performance compared to RNN, with an accuracy of 0.824 and higher precision, recall, and F1-score values of 0.800, 0.783, and 0.789, respectively. Notably, Bert-Large, leveraging the BERT language model, outperforms both RNN and CNN, achieving the highest accuracy of 0.845 and demonstrating robust precision, recall, and F1-score values of 0.816, 0.827, and 0.821, respectively. Furthermore, Bert-Large exhibits a Cohen's Kappa coefficient of 0.723 and an Area Under the Curve (AUC) value of 0.867, indicating good agreement beyond chance and excellent discriminative ability between classes. Overall, these results underscore the effectiveness of deep learning approaches, particularly Bert-Large, in sentiment analysis tasks, showcasing their potential for real-world applications in natural language processing.

C. VALIDATION WITH DIFFERENT DATASETS

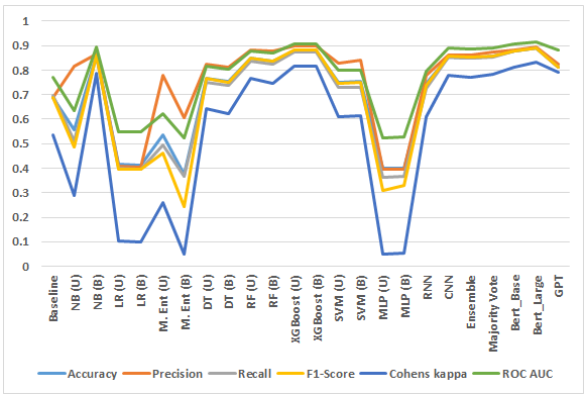
Three case studies are used to validate the trained models developed in Section IV. A publicly available labeled sentiment dataset related to software engineering is chosen for each case study, and trained models are used to predict this new dataset. Six evaluation metrics are used to evaluate prediction results.

1) CASE STUDY -1: STACK OVERFLOW DATASET

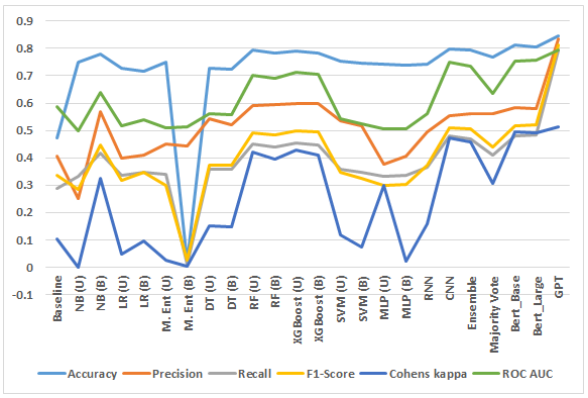
Stack Overflow is a programming-specific community question-answering website. Calefato et al, [18], created a sentiment dataset of 4423 Stack Overflow posts. This dataset includes questions, answers, and comments. Positive, negative, and neutral are labels given to each post. The dataset contains 1525 positive posts, 1204 negative posts, and 1694 neutral posts.

The pre-trained classifiers are applied to the Stack Overflow dataset and classification results are obtained. Figure 4 shows the results of 24 experiments. The Bert-Large model has a higher accuracy of 0.888 when compared to the other classifiers. The precision of XGBoost using bigram features is 0.898, while that of the Bert-Large model is 0.893, which is slightly lower than that of XGBoost. The Bert-Large model has a recall of 0.889, which is higher than all other models. The Bert-Large model has an F1-score of 0.890, which is higher than all other models. Similarly, the Cohen Kappa of the Bert-Large model is 0.831, which is higher than the Cohen Kappa of all models. Finally, the AUC of the Bert-Large model is 0.916, which is higher than the AUC of all other models. Based on these findings, one can

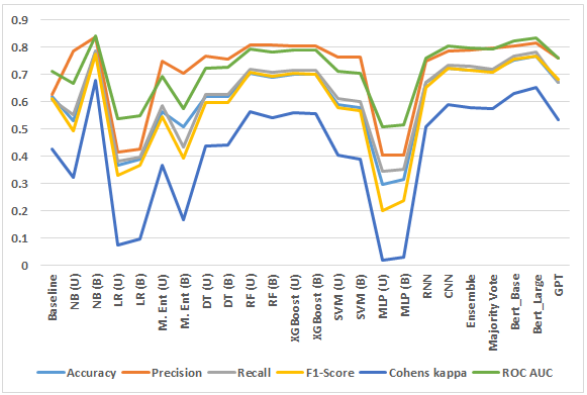
FIGURE 3. 4 layer CNN model.



(a) Classification Results on Stack Overflow Dataset



(b) Classification Results on JavaLib Dataset



(c) Classification Results on Jira Dataset

FIGURE 4. Classification results on different datasets.

conclude that the Bert-Large model produces the best results for classifying the Stack Overflow sentiment dataset.

2) CASE STUDY -2: JAVALIB DATASET

Lin et al [15] created the JavaLib dataset. This dataset is derived from Stack Overflow and consists of sentences extracted at random from JavaLib posts. Five researchers labeled the JavaLib dataset as positive, negative, and neutral. The labeled JavaLib dataset can be obtained from [20]. This dataset includes 1500 posts. The dataset contains 131 positive posts, 178 negative posts, and 1191 neutral posts.

On the Java dataset, results of pre-trained classifiers are obtained. Figure 4b shows the results of 24 experiments. The accuracy of the GPT model is 0.846, which is higher than the accuracy of the other classifiers.

TABLE 6. Comparison with state of the art tools for domain-specific sentiment analysis.

Dataset	Technique	Precision	Recall	F-Score
Jira	SentiStrength-SE	0.774	0.822	0.797
	Senti4SD	0.795	0.799	0.797
	SentiCR	**0.820**	**0.800**	**0.800**
	NB (B)	0.873	0.786	0.775
	XGBoost	0.806	0.717	0.703
	Bert-Large	**0.817**	**0.781**	**0.769**
SO	SentiStrength-SE	0.800	0.800	0.800
	Senti4SD	0.860	0.870	0.860
	SentiCR	0.820	0.810	0.820
	NB (B)	0.867	0.859	0.862
	XGBoost	0.897	0.875	0.882
	Bert-Large	**0.893**	**0.889**	**0.890**

3) CASE STUDY -3: JIRA DATASET

Jira is an issue-tracking system where developers can post issues and comments. Ortu et al. [41] created the Jira dataset,

TABLE 7. Qualitative comparison between the bert-large and existing techniques for sentiment analysis.

Technique	Methodology	Pre-Processing	Complexity
SentiStrength-SE	Qualitative and Quantitative	Remove Numbers, URL, Code Snippet, Stack Trace and Convert to Lowercase	$O(n^3)$
Senti4SD	Qualitative and Quantitative	Remove URL, HTML Tags and Code snippet	$O(n^3)$
SentiCR	Qualitative and Quantitative	Remove Numbers, URL, Code Snippet, Stack Trace, Convert to Lowercase and Emoji	$O(tdxlogn)$
Proposed	Quantitative	Remove Numbers, URL, Code Snippet, Stack Trace, Convert to Lowercase and Emoji	$O(n^2d + nd^2)$

which consists of issue comments posted by developers about popular open-source software such as Apache, Spring, Jboss, and CodeHaus. Jira dataset is available for download from [20] and it is used to validate the classifiers. This dataset contains 2565 issue comments. There are 1103 positive comments, 760 negative comments, and 702 neutral comments.

Figure 4c shows classification results from pre-trained classifiers on the Jira dataset. On the Jira dataset, the Na"ive Bayes classifier with bigram features outperformed all other classifiers. The NB model's accuracy is 0.788, which is 2% higher than the Bert-Large model. The precision of the NB model is 0.837, which is also 2% higher than the Bert-Large model. The recall of NB is 0.786, which is close to the recall of the Bert-Large model (0.781). The F1-score of NB is 0.775, which is also close to the F1-score of the Bert-Large model (0.769). Similarly, the Cohen Kappa of NB is 0.680, which is higher than the Cohen Kappa of all models. Finally, the AUC of NB is 0.843, which is higher than the AUC of all models. From these results, one can say that the NB model gives the best results to classify the Jira sentiment dataset.

VI. COMPARISON WITH STATE-OF-THE-ART

In this Section, the performance of the best sentiment analysis classification models are identified and compared. To provide a comprehensive benchmark, trained models are compared with three state-of-the-art tools commonly used in sentiment analysis within the software engineering domain: SentiStrength-SE [42], Senti4SD [18], and SentiCR [43]. The comparison utilized two benchmark datasets, Jira and Stack Overflow since these tools are specifically trained on these datasets. The evaluation was based on essential sentiment analysis metrics, including precision, recall, and F1-score. Detailed evaluation results are presented in Table 6.

Notably, findings indicate that the trained classification models consistently outperform all existing state-of-the-art tools on the Jira and Stack Overflow datasets. Particularly on the Jira dataset, the results of SentiCR and the Bert-Large model surpassed other trained classification models and state-of-the-art tools. F1-score results reveal a substantial 9% improvement of the Bert-Large model over SentiStrength-SE and Senti4SD. On the Stack Overflow dataset, the Bert-Large model exhibited superior performance over all existing state-of-the-art tools and trained classification models. These findings emphasize that the Bert-Large model, in particular, excels in sentiment analysis within the software engineering domain. It provides a notable improvement over current state-of-the-art tools, showcasing its potential to enhance sentiment analysis processes in this specific domain. Further, the models demonstrate a capacity to tackle domain-specific challenges and deliver superior results. While these findings are encouraging, it is essential to recognize that every model has its limitations, and these should be considered in practice and future research.

A. QUALITATIVE COMPARISON

This section provides a qualitative comparison of the proposed sentiment analysis scheme to many existing techniques, including SentiStrength-SE, Senti4SD, and SentiCR. The assessment is based on methodology, preprocessing, and time complexity. The qualitative comparison is given in Table 7.

The suggested scheme takes a quantitative approach to sentiment analysis, emphasizing mathematical modeling and statistical analysis of textual data. SentiStrength-SE, Senti4SD, and SentiCR, on the other hand, use both qualitative and quantitative approaches, including linguistic rules, sentiment lexicons, and machine learning algorithms. While proposed methodology provides a more systematic and rigorous research framework, existing techniques' hybrid methodologies may provide for greater flexibility in dealing with varied data sources and linguistic variances.

In terms of preprocessing, the proposed approach is comparable to existing strategies, with the removal of numerals, URLs, code snippets, and conversion to lowercase. SentiCR and the suggested approach also handle emojis as part of their preprocessing workflow. Emoji handling may increase sentiment analysis accuracy, especially in circumstances where emojis are used extensively to indicate emotions.

One noticeable distinction is the complexity time of each method. SentiStrength-SE and Senti4SD have a temporal complexity of $O(n^3)$, which indicates a cubic growth rate in computing resources as input size grows. SentiCR's temporal complexity model is $O(tdxlogn)$, where t represents iterations, d denotes dimensionality, and n represents input size. The proposed strategy has a temporal complexity of $O(n^2d+nd^2)$, providing a more efficient and scalable solution than existing strategies.

VII. CONCLUSION AND FUTURE WORK

In this research, a comprehensive large-scale study was undertaken, with the primary objective of improving

sentiment analysis accuracy in the domain of software engineering by using transformers. The study involved training various sentiment analysis classifiers and evaluating their performance against existing sentiment analysis tools in the field. Notably, an extensive pre-processing stage was employed to enhance the accuracy of the classifiers. The selection of the SentiSE dataset, derived from diverse software engineering interactions, as the training data, reflects a commitment to capturing the nuances of sentiment within the domain. Among the various models explored, including XGBoost, Bert-Base fine-tuned model, Bert-Large, and GPT fine-tuned model, the results underscored the exceptional performance of Bert-Large across multiple datasets. For instance, the Bert-Large model achieved an accuracy of 0.806 on the JavaLib dataset, 0.766 on the Jira dataset, and 0.888 on the Stack Overflow dataset. It is worth noting that these results exhibit a remarkable level of accuracy, particularly in challenging domains such as software engineering. An insightful aspect of the evaluation is that the Bert-Large model surpasses all other existing state-of-the-art sentiment analysis tools on both the Jira and Stack Overflow datasets. This not only signifies the effectiveness of the Bert-Large model but also underscores its potential as the go-to tool for sentiment analysis in these specific domains. Furthermore, the Bert-Large model exhibits comparable performance to SentiCR on the Jira dataset, while surpassing Senti4SD, a model trained specifically on the Stack Overflow dataset. These results emphasize the versatility and superiority of the Bert-Large model across a spectrum of software engineering datasets. In summary, the findings of this research not only provide evidence of significant improvements in sentiment analysis accuracy within the software engineering domain but also establish the Bert-Large model as a robust and superior choice when compared to existing state-of-the-art sentiment analysis tools, particularly on Jira and Stack Overflow datasets.

Efforts are underway to improve the accuracy of fine-tuned Bert-based classifiers for sentiment analysis of software domains. For this purpose class balancing and data smoothing techniques will be used.

DECLARATION OF INTERESTS

The authors affirm that they have no known financial or interpersonal conflicts that would have seemed to have an impact on the research presented in this study.

ACKNOWLEDGMENT

The authors would like to thank all of their colleagues who motivated and helped them to complete this research. Special thanks to all the anonymous reviewers who are working day and night to facilitate the researchers by providing valuable feedback to improve the quality of the research.

REFERENCES

[1] A. D'Andrea, F. Ferri, P. Grifoni, and T. Guzzo, "Approaches, tools and applications for sentiment analysis implementation," *Int. J. Comput. Appl.*, vol. 125, no. 3, pp. 26–33, Sep. 2015.

[2] R. Feldman, "Techniques and applications for sentiment analysis," *Commun. ACM*, vol. 56, no. 4, pp. 82–89, Apr. 2013.

[3] M. Obaidi and J. Klünder, "Development and application of sentiment analysis tools in software engineering: A systematic literature review," in *Proc. Eval. Assessment Softw. Eng.*, 2021, pp. 80–89.

[4] Z. Anwar and H. Afzal, "Mining crowd sourcing repositories for open innovation in software engineering," *Automated Softw. Eng.*, vol. 31, no. 1, p. 11, May 2024.

[5] K. Labille, S. Gauch, and S. Alfarhood, "Creating domain-specific sentiment lexicons via text mining," in *Proc. Workshop Issues Sentiment Discovery Opinion Mining (WISDOM)*, 2017, pp. 1–8.

[6] M. U. S. Khan, A. Abbas, A. Rehman, and R. Nawaz, "HateClassify: A service framework for hate speech identification on social media," *IEEE Internet Comput.*, vol. 25, no. 1, pp. 40–49, Jan. 2021.

[7] S. Vohra and J. Teraiya, "A comparative study of sentiment analysis techniques," *J. JIKRCE*, vol. 2, no. 2, pp. 313–317, 2013.

[8] S. Mohammad, M. U. S. Khan, M. Ali, L. Liu, M. Shardlow, and R. Nawaz, "Bot detection using a single post on social media," in *Proc. 3rd World Conf. Smart Trends Syst. Secur. Sustainablity (WorldS4)*, Jul. 2019, pp. 215–220.

[9] R. S. Jagdale, V. S. Shirsat, and S. N. Deshmukh, "Sentiment analysis on product reviews using machine learning techniques," in *Cognitive Informatics and Soft Computing*. New York, NY, USA: Springer, 2019, pp. 639–647.

[10] V. S. Shirsat, R. S. Jagdale, and S. N. Deshmukh, "Sentence level sentiment identification and calculation from news articles using machine learning techniques," in *Computing, Communication and Signal Processing*. New York, NY, USA: Springer, 2019, pp. 371–376.

[11] M. Guia, R. R. Silva, and J. Bernardino, "Comparison of Naïve Bayes, support vector machine, decision trees and random forest on sentiment analysis," in *Proc. 11th Int. Joint Conf. Knowl. Discovery, Knowl. Eng. Knowl. Manage. (IC3K)*. Setúbal, Portugal: SCITEPRESS, 2019, pp. 525–531.

[12] B. Gaye and A. Wulamu, "Sentimental analysis for online reviews using machine learning algorithms," *Int. Res. J. Eng. Technol. (IRJET)*, vol. 6, no. 8, pp. 1270–1275, 2019.

[13] M. Kabir, M. M. J. Kabir, S. Xu, and B. Badhon, "An empirical research on sentiment analysis using machine learning approaches," *Int. J. Comput. Appl.*, vol. 43, no. 10, pp. 1–9, Jul. 2019.

[14] J. KapočiūtDzikien, R. Damaševičius, and M. Woźniak, "Sentiment analysis of Lithuanian texts using traditional and deep learning approaches," *Computers*, vol. 8, no. 1, p. 4, Jan. 2019.

[15] 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 *Proc. IEEE/ACM 40th Int. Conf. Softw. Eng. (ICSE)*, May 2018, pp. 94–104.

[16] M. R. Islam and M. F. Zibran, "A comparison of dictionary building methods for sentiment analysis in software engineering text," in *Proc. ACM/IEEE Int. Symp. Empirical Softw. Eng. Meas. (ESEM)*, Nov. 2017, pp. 478–479.

[17] N. Novielli, D. Girardi, and F. Lanubile, "A benchmark study on sentiment analysis for software engineering research," in *Proc. IEEE/ACM 15th Int. Conf. Mining Softw. Repositories (MSR)*, May 2018, pp. 364–375.

[18] F. Calefato, F. Lanubile, F. Maiorano, and N. Novielli, "Sentiment polarity detection for software development," *Empirical Softw. Eng.*, vol. 23, no. 3, pp. 1352–1382, Jun. 2018.

[19] M. R. Islam and M. F. Zibran, "A comparison of software engineering domain specific sentiment analysis tools," in *Proc. IEEE 25th Int. Conf. Softw. Anal., Evol. Reengineering (SANER)*, Mar. 2018, pp. 487–491.

[20] A. Bosu. (2018). *Sentise is a Sentiment Analysis Tool for Software Engineering Interactions*. [Online]. Available: https://github.com/amiangshu/SentiSE/blob/master/models/sentise-oracle1.xlsx

[21] S. Alam and N. Yao, "The impact of preprocessing steps on the accuracy of machine learning algorithms in sentiment analysis," *Comput. Math. Org. Theory*, vol. 25, no. 3, pp. 319–335, Sep. 2019.

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

[23] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," OpenAI, Tech. Rep., 2018.

[24] C. McCormick and N. Ryan. (Jul. 22, 2019). *Bert Fine-Tuning Tutorial With Pytorch*. [Online]. Available: http://www.mccormickml.com
[25] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Nov. 2011.
[26] Z. Jianqiang, G. Xiaolin, and Z. Xuejun, "Deep convolution neural networks for Twitter sentiment analysis," *IEEE Access*, vol. 6, pp. 23253–23260, 2018.
[27] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," 2014, *arXiv:1409.2329*.
[28] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proc. Conf. empirical methods natural Lang. Process. (EMNLP)*, 2014, pp. 1532–1543.
[29] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, and J. Brew, "HuggingFace's transformers: State-of-the-art natural language processing," 2019, *arXiv:1910.03771*.
[30] A. McCallum and K. Nigam, "A comparison of event models for naive Bayes text classification," in *Proc. AAAI Workshop Learn. Text Categorization*, 1998, vol. 752, no. 1, pp. 41–48.
[31] A. Genkin, D. D. Lewis, and D. Madigan, "Large-scale Bayesian logistic regression for text categorization," *Technometrics*, vol. 49, no. 3, pp. 291–304, Aug. 2007.
[32] K. Nigam, J. Lafferty, and A. McCallum, "Using maximum entropy for text classification," in *Proc. IJCAI Workshop Mach. Learn. Inf. filtering*, Stockholm, Sweden, 1999, vol. 1, no. 1, pp. 61–67.
[33] E. Loper and S. Bird, "NLTK: The natural language toolkit," 2002, *arXiv:0205028*.
[34] J. R. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, Mar. 1986.
[35] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, pp. 5–32, Oct. 2001.
[36] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 785–794.
[37] T. Joachims, *Learning to Classify Text Using Support Vector Machines*, vol. 668. Berlin, Germany: Springer, 2002.
[38] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE Trans. Neural Netw.*, vol. 13, no. 2, pp. 415–425, Mar. 2002.
[39] F. Murtagh, "Multilayer perceptrons for classification and regression," *Neurocomputing*, vol. 2, nos. 5–6, pp. 183–197, Jul. 1991.
[40] Z. Anwar, H. Afzal, A. Ahsan, N. Iltaf, and A. Maqbool, "A novel hybrid CNN-LSTM approach for assessing StackOverflow post quality," *J. Intell. Syst.*, vol. 32, no. 1, Nov. 2023, Art. no. 20230057.
[41] M. Ortu, A. Murgia, G. Destefanis, P. Tourani, R. Tonelli, M. Marchesi, and B. Adams, "The emotional side of software developers in JIRA," in *Proc. 13th Int. Conf. Mining Softw. Repositories*, May 2016, pp. 480–483.
[42] M. R. Islam and M. F. Zibran, "Leveraging automated sentiment analysis in software engineering," in *Proc. IEEE/ACM 14th Int. Conf. Mining Softw. Repositories (MSR)*, May 2017, pp. 203–214.
[43] T. Ahmed, A. Bosu, A. Iqbal, and S. Rahimi, "SentiCR: A customized sentiment analysis tool for code review interactions," in *Proc. 32nd IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE)*, Oct. 2017, pp. 106–111.

ZEESHAN ANWAR is currently pursuing the Ph.D. degree in software engineering with the Department of Computer Software Engineering, National University of Science and Technology (NUST), Islamabad, Pakistan. He is working on Text Mining under the supervision of Dr. Hammad Afzal. He has various publications in international journals and conferences. His research interests include software engineering, software testing, project management, text mining, and computational intelligence.

HAMMAD AFZAL received the Ph.D. degree from The University of Manchester, U.K., in December 2009. He was affiliated with the Digital Enterprise Research Institute (DERI), National University of Ireland, Galway, as a Research Intern, from July 2009 to December 2009. He is currently a Professor of computer science and artificial intelligence with the National University of Sciences and Technology (NUST), Pakistan. His main research interests include natural language processing and computational semantics text mining.

TAHER AL-SHEHARI received the B.Sc. degree (Hons.) in computer science from King Khalid University, in 2007, and the M.S. degree in computer science from the King Fahd University of Petroleum and Minerals, in 2014. He is currently an IT Lecturer with King Saud University. His general research area is information security and privacy. His research interests include traffic analysis attacks, anonymity systems, data analysis, and website fingerprinting.

MUNA AL-RAZGAN received the Ph.D. degree in information technology from George Mason University, Fairfax, VA, USA. She is currently an Associate Professor in software engineering with the College of Computer and Information Sciences, King Saud University, Riyadh, Saudi Arabia. Her research interests include data mining, machine learning, artificial intelligence, educational data mining, and assistive technologies.

TAHA ALFAKIH received the B.S. degree in computer science from the Department of Computer Science, Hadhramout University, Yemen, the M.Sc. degree from the Department of Computer Science, King Saud University (KSU), Riyadh, Saudi Arabia, and the Ph.D. degree from the Department of Information System, KSU. He is currently a Researcher with the Computer Science College, KSU. His research interests include machine learning, mobile edge computing, and the Internet of Things (IoT).

RAHEEL NAWAZ is currently the Pro-Vice-Chancellor of Staffordshire University, U.K., and also a leading Researcher in artificial intelligence and digital education. He holds several adjunct professorships and scientific directorships across Asia and North America. He sits on the boards of research and charitable organizations, such as the National Centre for Artificial Intelligence (Pakistan), TechSkills (U.K.), and NTF (U.K.), and has advised national policy organizations including the Prime Minister's Task Force on Science and Technology (Pakistan). He has authored more than 150 peer-reviewed research articles and his career grant capture stands at more than 14 million. He has graduated 19 Ph.D. students so far. According to Google Scholar, he is among the top-10 most cited scholars in the world in the fields of digital transformations, applied artificial intelligence, and educational data science.