# Incorporating Pre-trained Transformer Models into TextCNN for Sentiment Analysis on Software Engineering Texts


Kexin Sun
*State Key Lab for Novel Software Technology*
*Nanjing University*
Nanjing, China
mf20320130@smail.nju.edu.cn

XiaoBo Shi
*College of Information Science and Technology*
*Dalian Maritime University*
Dalian, China
xiaobo0528n@gmail.com

Hui Gao
*State Key Lab for Novel Software Technology*
*Nanjing University*
Nanjing, China
ghalexcs@gmail.com

Hongyu Kuang*
*State Key Lab for Novel Software Technology*
*Nanjing University*
Nanjing, China
khy@nju.edu.cn

Xiaoxing Ma
*State Key Lab for Novel Software Technology*
*Nanjing University*
Nanjing, China
xxm@nju.edu.cn

Guoping Rong
*State Key Lab for Novel Software Technology*
*Nanjing University*
Nanjing, China
ronggp@nju.edu.cn

Dong Shao
*State Key Lab for Novel Software Technology*
*Nanjing University*
Nanjing, China
dongshao@nju.edu.cn

Zheng Zhao
*College of Artificial Intelligence at Dalian Maritime University*
Dalian, China
zhaozheng@dlmu.edu.cn

He Zhang
*State Key Lab for Novel Software Technology*
*Nanjing University*
Nanjing, China
hezhang@nju.edu.cn



## ABSTRACT

Software information sites (e.g., Jira, Stack Overflow) are now widely used in software development. These online platforms for collaborative development preserve a large amount of Software Engineering (SE) texts. These texts enable researchers to detect developers' attitudes toward their daily development by analyzing the sentiments expressed in the texts. Unfortunately, recent works reported that neither off-the-shelf tools nor SE-specified tools for sentiment analysis on SE texts can provide satisfying and reliable results. In this paper, we propose to incorporate pre-trained transformer models into the sentence-classification oriented deep learning framework named TextCNN to better capture the unique expression of sentiments in SE texts. Specifically, we introduce an optimized BERT model named RoBERTa as the word embedding layer of TextCNN, along with additional residual connections between RoBERTa and TextCNN for better cooperation in our training framework. An empirical evaluation based on four datasets from different software information sites shows that our training framework can achieve overall better accuracy and generalizability than the four baselines.



*Hongyu Kuang is the corresponding author.




*Internetware 2022, June 11–12, 2022, Hohhot, China*



ACM ISBN 978-1-4503-9780-3/22/06…$15.00
https://doi.org/10.1145/3545258.3545273



## KEYWORDS

Sentiment Analysis, Pre-trained Models, Software Mining, Nature Language Processing

**ACM Reference Format:**
Kexin Sun, XiaoBo Shi, Hui Gao, Hongyu Kuang, Xiaoxing Ma, Guoping Rong, Dong Shao, Zheng Zhao, and He Zhang. 2022. Incorporating Pre-trained Transformer Models into TextCNN for Sentiment Analysis on Software Engineering Texts . In *13th Asia-Pacific Symposium on Internetware (Internetware 2022), June 11–12, 2022, Hohhot, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3545258.3545273


## 1 INTRODUCTION

Modern software development is now a collaborative activity that often requires developers to coordinate and communicate with their team members, or other developers from the community. These intense interactions help developers to achieve both a healthy status and a satisfying conclusion for their projects [46]. Thus, as the Internet has become the indispensable infrastructure for exchanging resources in most domains, software information sites play a vital role for software developers to search for solutions, share experiences, offer help, and learn new techniques [3, 4, 17, 53]. These sites include online development tools (e.g., GitHub and Jira), and community-based Q&A platforms (e.g., Stack Overflow and app stores). According to the recent reports (at the end of 2021)[1] [2], these software information sites have formed collaborative social networks with vast numbers of users (e.g., 73 million developers on GitHub, and 100 million monthly visitors to Stack Overflow),

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

and also continuously preserve a massive number of developer-generated texts (e.g., 61 million new repositories created in 2021 on GitHub, and 21 million questions asked to-date on Stack Overflow) about the daily development process for software engineering (SE). Therefore, these SE texts offer a great opportunity for researchers to find out whether the developers are satisfied or frustrated about their ongoing work of software development, i.e., their positive, neutral, or negative sentiments expressed in the SE texts, without hindering them by observations or interactions required in the traditional analyses [35, 55, 56]. Thus, to further support software engineering tasks such as development, maintenance, and evolution, a growing body of proposed approaches adopts sentiment analysis on SE texts collected from the software information sites such as GitHub [19, 42, 47], JIRA [33, 39], Stack Overflow [6, 29, 45], and app stores [20, 54]. Recent research further shows that, because of the wide availability of development-related SE texts, the sentiment analysis on SE texts has become a vital way to inspect the statuses of software development communities, such as predicting community smells' occurrence on individual developers [23], or tracing self-admitted technical debts to the code [16].

Unfortunately, recent works reported that the results of analyzing sentiments from SE texts are unsatisfying and unreliable when using off-the-shelf sentiment analysis tools [25], or even SE-specified ones [29]. The reason why sentiment analysis on SE texts continues to be challenging is that, unlike the texts from movie comments [48] or general social network [52] where off-the-shelf tools are built, the sentiment expression in SE texts is more indirect and dispersed [50] because the developers often have to first describe the issues that they faced or proposed in their development work, then to express their sentiments on the described issues. This situation raises two specific challenges for sentiment analysis on SE texts: (1) the likely misclassification of neutral technical texts in the SE domain as sentimentally negative (e.g., "kill", "critical", "exception") [38]; (2) the complicated sentence structure with multiple clauses that are actually not sentimental [50].

One possible direction for researchers to meet these two challenges is to propose heuristic-based approaches that build a SE-specified dictionary [24] or filter-adjust rules for mainstream sentiment dictionaries [50]. Although being able to provide more insights and thus achieve generalizable results on different datasets, the improvement brought by these approaches is limited to the depth of the researchers' manual explorations. Another direction, which is the mainstream in this field, is to turn sentiment analysis into a classification task of machine learning on sentimentally-labeled SE texts [5, 24, 52]. However, Lin et al. [29] recently reported that their proposed learning-based approach, which is adopt-ed from the Recursive Neural Network (RNN) model of Stanford CoreNLP on their SE-specified training set (containing 1500 texts elicited from Stack Overflow), did not significantly outperform off-the-shelf sentiment analysis tools. Meanwhile, Calefato et al. [5] proposed a learning-based approach named Senti4SD to improve sentiment analysis on SE texts from three aspects: (1) they build a larger training set containing 4423 texts mined from Stack Overflow; (2) they use word2vec [36] to build a specified Distributional Semantic Model (DSM) over 20 million texts from Stack Overflow; (3) based on the DSM, they use the Support Vector Machine (SVM) technique to train their training set based on a suite of lexicon-based, keyword-based, and semantic features. Senti4SD significantly outperforms SentiStrength [52] (the widely used sentiment analysis tool) on the authors' mined training set. However, Sun et al. [50] reported that the generalizability of Senti4SD is not satisfying, i.e., Senti4SD cannot outperform other sentiment analysis tools on three different, yet widely-researched datasets.

To better support the common software development in practice, we argue that the generalizability of a SE-specified sentiment analysis tool is also very important as its overall accuracy, because the SE texts can be extracted from different software information sites and relate to various software projects where few SE texts are labeled with sentiments for re-training, and the writing of these SE texts can be obviously different. In this regard, we propose a training framework by incorporating pre-trained transformer models into TextCNN [26], which is a widely-used Convolutional Neural Network (CNN) model for sentence classification, to carry out the sentiment polarity classification task on SE texts collected from software information sites. Our goal is to improve the accuracy of learning-based sentiment analysis on SE texts, along with improved generalizability on different datasets where the SE texts are mined from different software information sites, to better support SE tasks. Zhang et al. [57] reported that using pre-trained transformer models (specifically, these models are used for word embedding) is beneficial for sentiment analysis for software engineering, but they did not propose an integrated training framework with both pre-trained transformer models and spliced downstream models specifically for sentiment classification task on SE texts.

In particular, our training framework consists of three parts. First, we use RoBERTa [31] (robustly optimized BERT approach [9]) as our word embedding layer. We choose RoBERTa as the incorporated pre-trained model because it is not only reported to obtain state-of-the-art performances on different Nature Language Process (NLP) benchmarks when released, but also removes the Next Sentence Prediction (NSP) loss that is designed for Natural Language Inference in BERT, thus likely to focus more on other downstream NLP tasks such as sentence classification. Second, we use TextCNN as the spliced model for our downstream, sentiment classification task on SE texts. TextCNN is designed for sentence classification by adopting the CNN architecture to explore the semantics from groups of adjacent words in each sentence. We expect this inherent exploration can help further capture the implicit sentiment expression in SE texts for our classification task. Meanwhile, we argue that the simple but concise structure of TextCNN is suitable to analyze SE texts because the writing of SE texts and software documentation is a typical technical-writing [7], which relies more on domain-specific words and is thus more condense compared to general texts of natural language. Finally, we introduce an additional residual connection into our framework to better cooperate with RoBERTa which is built upon a much larger dataset and has many parameters to fine-tune. We first trained and evaluated our framework on the same training set with 4423 Stack Overflow posts from Senti4SD. The result shows that our trained model outperforms Senti4SD in the test set. We then added three different datasets to further evaluate our same trained model with four baseline approaches [5, 24, 50, 52]. The result shows that our trained model can

achieve both good accuracy and generalizability on four datasets created from different software information sites.

This paper aims to improve sentiment analysis for software engineering by incorporating pre-trained transformer models into the CNN architecture to carry out sentiment polarity classification tasks on SE texts. We name our approach as **EASTER** (s**E**ntiment **A**nalysis on **S**E texts based on **TE**xtcnn and **R**oBERTa). This paper makes two contributions: (1) we proposed an integrated training framework that incorporates RoBERTa into TextCNN; (2) we set up evaluations based on four datasets and show that our trained model can achieve both good accuracy and generalizability when analyzing sentiments on SE texts. Our training framework, trained model, and datasets for replication are publicly available [13].

## 2 RELATED WORK

### 2.1 Pre-trained Transformer Models

Pre-training provides a better model initialization, which usually leads to better generalization performance and faster convergence on the target task [43], and can also help to avoid overfitting on small data [14]. Qiu et al. [43] categorized pre-trained models for NLP into two generations. The first generation of pre-trained models aims to learn word embeddings, such as Skip-Gram [37] and GloVe (Global Vectors for Word Representation) [40]. The second-generation focuses on learning contextual word embeddings, such as CoVe (Contextualized Word Vectors) [34], ELMo (Embeddings from Language Models) [41], and BERT [9]. An increasing number of SE studies adopted pre-training models. Sulistya et al. [49] using word embeddings to identify software-related tweets. Lin et al. [30] proposed T-BERT (Trace BERT) based on BERT to generate traceability links. Zhang et al. [57] reported that pre-trained transformer models are beneficial for sentiment analysis on SE. But they did not propose an integrated training framework with both pre-trained transformer models and spliced downstream models specifically for the sentiment classification task on SE texts, while we proposed the integrated training framework named EASTER.

### 2.2 Learning-based Sentiment Analysis for SE

Senti4SD [5] is a supervised learning-based tool based on the set of questions, answers, and comments on 4K Stack Overflow data. It uses three kinds of features when conducting sentiment classification tasks, i.e., dictionary-based features (i.e., the dictionary used by SentiStrength), keyword-based features (i.e., uni-grams and bi-grams extracted from large scale Stack Overflow posts), and semantic features (based on the word embeddings trained on Stack Overflow posts). SentiCR [1] is a supervised tool trained on Gradient Boosting Tree (GBT) that is specially designed for code review comments. It generates feature vectors by computing TF-IDF values for extracted bag-of-words from the input text. SentiSW [11] applies 6 supervised machine learning techniques (e.g., Random Forest (RF), Bootstrap Aggregating (Bagging), GBT, Naïve Bayes (NB), Ridge Regression and Linear Support Vector Machine (SVC)) to detect the sentiment of issue comments at entity-levels. In comparison, to the best of our knowledge, EASTER is the first integrated framework that tries to take advantage of both pre-trained transformer models and Deep Neural Networks (i.e., TextCNN) to better analyze sentiments specifically on SE texts.

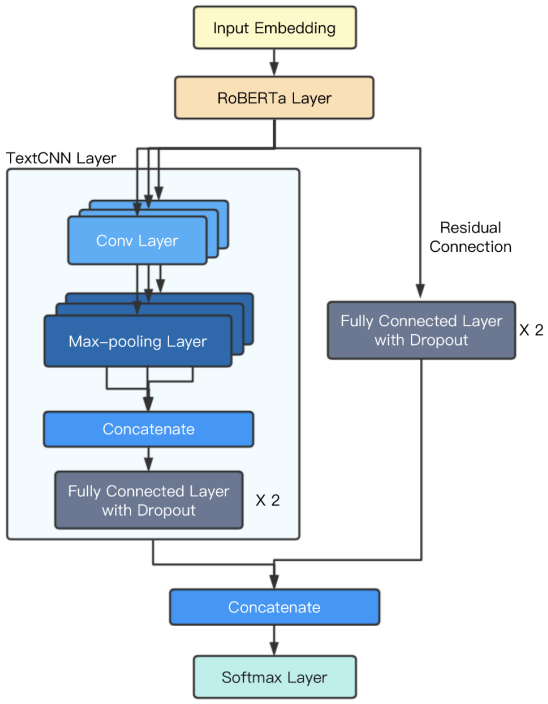


**Figure 1: The construction of EASTER**

### 2.3 Heuristic-based Sentiment Analysis for SE

SentiStrenght-SE [24] is a customized version of the mainstream dictionary-based approach called SentiStrength, implemented by replacing the original dictionary with a domain-specific one, which is the first SE-specific sentiment analysis. Through manual tagging, they obtained a SE-specific sentiment dictionary composed of 167 positive wildcards and 293 negative wildcards from 5992 JIRA question comments. In addition, they also use regular expressions to remove code fragments, such as web addresses, IP, etc. during preprocessing. SESSION [50] proposed a set of filter-adjust rules based on sentence structures (e.g., identifying subjunctive clauses or distinguishing the meaning of polysemous words) of SE texts, and combined these heuristics with SentiStrength. These two heuristic-based approaches can outperform baseline approaches on SE texts from their experiments, and also provide more insights and thus achieve generalizable results on different datasets, but their improvement is limited to the depth of manual investigation. The insights brought by the discussed works help us to propose to incorporate RoBERTa into TextCNN, so that our training framework can better capture the unique expression of sentiments in SE texts.

## 3 APPROACH

In this section, We discuss our training framework with RoBERTa [32] for upstream word embedding and TextCNN [26] for the downstream sentiment classification task. To make better cooperation between upstream and downstream, a residual connection is added. In the following subsections, we will explain each part of our model and the reasons for selection in more detail. The overview of our training framework is depicted in Figure 1.

## A Word embedding based on RoBERTa

We first briefly introduce pre-trained transformer models as the basis of following discussions. The language model represents a probability distribution over a word sequence, which with proper training can effectively capture the semantics of individual words based on their surrounding context. Transformer models can effectively learn the word distributions. It uses attention mechanisms to transduce the originally long embeddings with sparse information into the shorter ones to preserve semantics as much as possible. BERT [10] is a typical transformer model that uses a "masked language model" (MLM). The MLM randomly masks some of the tokens, and the goal is to predict the original vocabulary id of the masked word based only on its context, to pretrain a deep bidirectional Transformer. Researches [22, 44] showed that a good pre-trained model can improve the generalizability of the trained model, and the speed of convergence in the training process with saved cost.

The pre-trained model RoBERTa, is a replication study of BERT to produce high-quality representations for the input texts, which are the basis of our downstream CNN layers. We choose RoBERTa because it has been optimized by enhanced training and the removal of the NSP loss. This optimization makes RoBERTa more suitable for our sentiment classification tasks on SE texts than other pre-trained models. In particular, RoBERTa trains the model on longer sequences and uses dynamic masking instead of static one. Longer sequences make it more suitable for the analysis task of longer texts, which corresponds to the observation of Sun et al. [50] that the sentiment expression in SE texts is more indirect and complicated due to the need of describing development-related issues. Unlike static masking, which only masks during data preprocessing, dynamic masking generates a new masking whenever a sequence is inputted into the model. This makes the masking different in each epoch and allows the model to better capture how sentiments are expressed in the sentence. Here, we selected a further fine-tuned version of RoBERTa in the study of Barbieri et al. [2]. They re-trained the model on 58M tweets and fine-tuned it for sentiment analysis with the TweetEval benchmark.

The input of our RoBERTa-based layer is the same as BERT. We use WordPiece embeddings with a 30,000-token vocabulary. A whole text is packed together into a single sequence. The first token of each sequence is mandated to be a special classification token ([CLS]) which will be used as an aggregate sequence representation for classification tasks in the final hidden state. Another special token ([SEP]) is used to separate different sentences from one text. The model uses segment embeddings to record which sentence the token belongs to, and position embeddings to record the index of the token in text. For a given token, its input representation is constructed by summing the corresponding token, segment, and position embeddings. A sample of generated input is depicted in Figure 2 [10]. The output of this layer is a $m \times n \times k$ matrix, where $m$ represents the number of sentences in a text, $n$ represents the number of word pieces in a sentence, and $k$ is a dimension of one word-piece embedding of the input SE text.

## B Classification based on TextCNN

In the research of Zhang et al. [58], they used the embeddings produced by the pre-trained transformer models (i.e. BERT, RoBERTa, XLNet, ALBERT) for sentiment classification in a straightforward manner. They simply added a feed-forward dense layer and the softmax activation function on top of each transformer model to find out whether these pre-trained models are beneficial to sentiment analysis for software engineering. The experimental results show that their trained models can outperform prior approaches (i.e. Stanford CoreNLP, SentiStrength, SentiStrength-SE, SentiCR, and Senti4SD), but there is still room for further improvement. In this paper, we use a CNN architecture specified for sentence classification (i.e., TextCNN [26]) to better fuse the semantic features provided by RoBERTa in the downstream sentiment classifications.

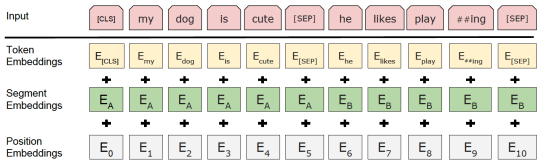


**Figure 2: The input representation of EASTER**

Specifically, a convolution kernel in CNN will go over each possible window and calculate a feature $c$ for them. For a kernel whose window size is $h$ and the stride is $s$, the calculation of $c$ can be described as follows. Let $x_i \in \mathbb{R}^k$ represent the $i$-th $k$-dimensional vector, $x_{i:i+j}$ refer to the concatenation of vectors $x_i$,$x_{i+1}$,...,$x_{i+j}$ . The feature $c_i$ can be expressed as $c_i = f(w \cdot x_{i:i+h-1} + b)$, where $w \in \mathbb{R}^{hk}$ is a convolution kernel, $b \in \mathbb{R}$ is a bias term and $f$ is a non-linear function such as the hyperbolic tangent. The sentence $\{x_{1:h}, x_{1+s:h+s}, ..., x_{n-h+1:n}\}$ can produce a feature map $c = [c_1, c_{1+s}, ..., c_{n-h+1}]$. Through the formula of a feature $c$, we can we can see that one feature fuses multiple information of input vectors, which makes CNN can have a greater vision and find longer informative clues.

Here, we choose TextCNN for the downstream task. It's a network trained on top of pre-trained word embeddings for sentence-level classification tasks and has a simple but concise structure, which consists of a convolutional layer, a max-pooling layer, and a fully connected softmax layer. Its convolutional layer uses multiple kernels to obtain multiple features. Each kernel has its window size and moves in a certain stride. Each possible window will be dot multiplied with the convolution kernel to produce feature maps. The specific calculation process is described above. In the max-pooling layer, a max-overtime pooling operation is applied over the feature map and takes the maximum value $c = max\{c\}$ as the feature corresponding to a particular filter. This operation can capture the most important feature for each feature map and naturally deals with variable sentence lengths. Next, the features will be passed to a fully connected softmax layer whose output is the probability distribution over labels. This layer employs dropout on the penultimate layer with a constraint on l2-norms of the weight vectors to prevent overfitting [21]. The construction of TextCNN can be seen in Figure 3 [26].

For the sentiment analysis task of SE texts, we further adjust the architecture of the network. We change the original single-channel network into three channels. Multi channels can make the model capture features from different angles as much as possible. Each

channel is convoluted and max-pooled independently. The output of them will be concatenated and then made a full connection with dropout twice. After this, the output of TextCNN will be concatenated with the output of the auxiliary classifier in the residual connection, and finally fed into the softmax layer.

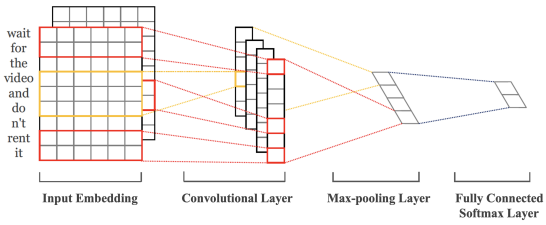


**Figure 3: The construction of TextCNN**

## C Residual connection between embedding and TextCNN

Residual connections have been used in many researches [28, 51] for addressing vanishing/exploding gradients.

In general, one layer of the neural network can be represented as $y = H(x)$, where $x$ and $y$ are considered as the input and output vectors of the layers, $H(x)$ is the mapping function between input and output. In a residual block, $H(x)$ can be represented as

$$H(x) = F(x, f\{W_i\}) \tag{1}$$

According to the formula, $F(x, \{W_i\}) = H(x) - x = y - x$, where $F(x, \{W_i\})$ represents the residual between input and output , and it also the mapping to be learned. This ensures the network of layer $i + 1$ to contain more antecedent information than layer $i$.

The upstream task of our model, i.e., RoBERTa, has a large number of parameters to be trained. To better fine-tune it, an additional head is added to connect both the RoBERTa layer and an additional auxiliary classifier. This classifier consists of two layers of full connection with dropout to preserve the antecedent information.

## 4 EXPERIMENTAL SETUP

We now introduce the experimental setup of our evaluation. Section 4.A shows the four evaluated datasets of SE texts. Section 4.B describes the training details of our model. Section 4.C defines metrics for evaluating our proposed approach. Section 4.D introduces our research questions and the design of experiments.

**Table 1: Datasets Used for Evalution**

| Dataset | texts | positive | neutral | negative |
|---|---|---|---|---|
| **Stack Overflow 4423** | 4423 | 1527 | 1694 | 1202 |
| **Stack Overflow 1500** | 1500 | 131 | 1191 | 178 |
| **App Reviews** | 341 | 186 | 25 | 130 |
| **JIRA Issues** | 926 | 290 | 0 | 636 |

## A The Benchmark

We first introduce the benchmark from the study of Lin et al. [29], which aims to find out whether current sentiment analysis tools can provide reliable results of sentiments expressed in the SE texts. It consists of three datasets that are built on 1500 Stack Overflow discussions, 341 app reviews, and 926 JIRA issues, respectively. In addition, we introduce another much larger dataset with 4423 Stack Overflow posts from the study of Calefato et al. [5], on which they proposed Senti4SD, one of the state-of-the-art learning-based approaches of sentiment analysis on SE texts. Table 1 reports the total number of texts, and the number of positive, neutral, and negative texts for each dataset, showing that the sentiment distributions in different datasets significantly vary (e.g., **Stack Overflow 1500** v.s. **JIRA Issues**). Thus, these datasets can additionally help to further evaluate how our trained model performs under different situations (i.e., *the generalizability*). In particular, we use the dataset of Senti4SD (i.e., **Stack Overflow 4423**) to train our model and the other three datasets to evaluate our trained model for evaluating the generalizability of our training framework, i.e., we did not re-train our model individually on the other three datasets.

## B Model Training

In this subsection, we will describe the hyperparameters and the hardware configurations used in training.

The upstream task of our model, RoBERTa, follows the architecture of $BERT_{LARGE}$ (the number of layers $L$ = 24, the hidden dimension $H$ = 1024, the number of self-attention heads $A$ = 16, 355M parameters) [10]. For the TextCNN Layer of our model, we set up three convolution channels, whose numbers of convolution kernels are 300, 300, and 200, the window sizes are 2, 2, and 3, and the strides are 2, 1, and 1 respectively. The number of neurons in the first fully connected layer in TextCNN layer is 150, the number of the second is 3, and their dropout rates are both 0.5. For the auxiliary classifier in the residual connection, the number of neurons in its first fully connected layer is 768, the number of the second is 3, and their dropout rates are both 0.1. When the loss doesn't decrease in 10 consecutive epochs, the model will early stop. This stopping condition can effectively prevent overfitting.

We conducted our experiment on 11th Gen Intel(R) Core(TM) i5-11500 @ 2.70GHz and 16GB RAM. We utilized 1 NVIDIA GeForce RTX 2080 Ti GPU with 11 GB memory to train and evaluate our model. Our model was implemented with Tensorflow v2.7.0 and HuggingFace v4.15.0. In the process of training and fine-tuning, we use Adam Optimizer [27] and set the initial learning rate as 1e-6. Because lengths of texts vary, in order to prevent errors in the convolution layer, we set the batch size to 1. Regarding the model selection, we split the **Stack Overflow 4423** into training(70%) and test(30%) sets, and perform 10-fold cross-validation on the training set to seek the optimal parameters of the model. The final model was trained on the whole training set using the optimal configuration and then evaluated on the test set and the other three datasets in the benchmark, to assess to what degree the model is able to generalize sentiment polarity classification on unseen new data. The reason why **Stack Overflow 4423** is selected as the training set is that this dataset has the largest number of text among our benchmark and its number of each polarity is relatively balanced.

**Table 2: The performance of Senti4SD and EASTER, Pure RoBERTa, and Pure TextCNN (all four of them are trained on 70% of Stack Overflow 4423) on the testing set (the other 30% of Stack Overflow 4423)**

| Tools | Overall Accuracy | Positive | | | Neutral | | | Negative | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | P | R | F | P | R | F | P | R | F |
| Senti4SD | 88.16% | 95.20% | 93.38% | 0.943 | 82.15% | 79.44% | 0.808 | 79.50% | 87.70% | 0.834 |
| EASTER | **92.53%** | 96.64% | **97.35%** | **0.967** | **90.05%** | 85.03% | **0.875** | 85.50% | **91.27%** | **0.883** |
| Pure RoBERTa | 86.73% | **97.83%** | 86.18% | 0.916 | 73.01% | **88.58%** | 0.800 | **86.35%** | 85.32% | 0.858 |
| Pure TextCNN | 81.37% | 94.46% | 85.29% | 0.896 | 73.88% | 75.38% | 0.746 | 65.16% | 80.16% | 0.719 |

## C Metrics

We use precision($P$), recall($R$), and F-measure($F$) to measure the performances of our trained model and baselines as follows:

$$P = \frac{|S_c \cap S_c^{'}|}{|S_c^{'}|}, \qquad R = \frac{|S_c \cap S_c^{'}|}{|S_c|}, \qquad F = \frac{2 \times P \times R}{P + R} \tag{2}$$

where $c$ represents the possible sentimental polarity of SE texts (i.e., positivity, negativity, and neutrality), $S_c$ represents the set of texts having $c$, $S_c^{'}$ represents the set of texts classified to have sentimental polarity $c$ by an approach. $F$ is the weighted harmonic mean of precision and recall. A higher $F$ means both $P$ and $R$ are high, and the tool performs better. Besides, we introduce the overall accuracy of sentimental analysis on the set $S$ for all of the three sentimental polarities as a metric. It can be calculated as follows:

$$Overall\ Accuracy \ = \ \frac{\sum_{c \in \text{polarities}} |S_c \cap S_c^{'}|}{|S|} \tag{3}$$

where we accumulate the numbers of texts in $S_c^{'}$, which have the same sentimental polarity "$c$" in $S_c^{'}$ for all three polarities, and then calculate the proportion of it in the given set $S$ of texts.

## D. Research Question

In this paper, we aim to study whether the approach incorporating pre-trained transformer models into the TextCNN can better perform in the sentiment analysis of SE texts. Furthermore, we also want to find out how many contributions can the pre-trained model RoBERTa and the spliced TextCNN for downstream tasks individually provide. Therefore, we propose the following three research questions:

*RQ1: Can our proposed approach outperform the baselines for analyzing sentiments on SE texts?*

*RQ2: How many contributions do RoBERTa make?*

*RQ3: How many contributions do TextCNN make?*

To study *RQ1*, we first compared the performance of trained models from **EASTER** and **Senti4SD** on the testing set (30% data of **Stack Overflow 4423**) of our study. In addition, to further evaluate the performance of the model trained under EASTER, we introduce the following four baselines of both heuristic (dictionary) -based and learning-based approaches: (1) **SentiStrength** [52], the state-of-the-art dictionary-based tool developed from the social networking texts; (2) **SentiStrength-SE** [24], a customized version of SentiStrength, implemented by replacing the original dictionary with a SE-specific one; (3) **SESSION** [50], a work based on SentiStrength, which uses filter and adjust rules based on sentence structures information to improve the sentiment analysis for SE texts; (4) **Senti4SD** [5], a representative SE-customized, learning-based tool that is trained on the **Stack Overflow 4423** dataset (also part of our evaluated datasets). It is worth noting that we re-trained Senti4SD on the same training set to explore its optimized performance. Our trained model achieves better overall accuracy and further discussion is in Section 5. We expect to find out whether EASTER performs better than the four baselines.

To study *RQ2*, we compare EASTER with the **Pure TextCNN**. We choose the CNN-rand version of TextCNN in Kim's paper [26], where all words are randomly initialized and then modified during training. We train the CNN-rand with our training set and evaluate its performance against our model on our testing set and benchmark. Through this comparison, we can see how many contributions can RoBERTa make to our model.

To study *RQ3*, we compare EASTER with the **Pure RoBERTa** which has already trained on 58M tweets and finetuned for sentiment analysis by Barbieri et al. [2]. We add a feed-forward dense layer and softmax activation function on top of it to get the classification results and further finetune it with our training set. We will evaluate its performance against EASTER on our testing set and the benchmark. Through this comparison, we can see how many contributions can TextCNN make to our model.

## 5 RESULTS AND DISCUSSIONS

### A. *RQ1: Can EASTER outperform the baselines?*

Table 2 shows the performances of EASTER and the original version of Senti4SD in the work of Lin et al. [5] on the testing set. Through the observation, we can find that EASTER's performance is significantly better than Senti4SD's. Its overall accuracy (92.53%) is 4.37% higher than Senti4SD 's (88.16%) and the F-measures of EASTER on the three sentimental polarities are all the highest in the table. The data proves that EASTER performs more excellent and comprehensive in our testing set.

Table 3 shows the performances of EASTER and the original version of Senti4SD, and the other three heuristic-based baseline approaches on the benchmark. It can be seen that EASTER can achieve the highest overall accuracy on the first three data sets (**Stack Overflow 4423, Stack Overflow 1500, App Reviews**) and the best values on the majority of F-measures. On **Stack Overflow 4423**, the overall accuracy of EASTER is only 1.77% higher than Senti4SD, because the testing set accounts for a small part of **Stack Overflow 4423**, as a result, the advantages of EASTER shrunk at the same time. It is worth noting that EASTER can still achieve more than 75% overall accuracy on **App Reviews** whose sentiment distribution is completely different from that of EASTER's training set (**App Reviews** are mainly composed of emotional texts,

**Table 3: The performances of SentiStrength, SentiStrength-SE, SESSION, the off-the-shelf model of Senti4SD provided by Calefato et al. [5], and EASTER (trained on 70% Stack Overflow 4423) on the four datasets**

| Dataset | Tool | Overall Accuracy | Positive | | | Neutral | | | Negative | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | P | R | F | P | R | F | P | R | F |
| Stack Overflow 4423 | SentiStrength | 81.55% | 88.90% | 92.34% | 0.906 | 92.76% | 63.58% | 0.754 | 66.83% | 93.18% | 0.778 |
| | SentiStrength-SE | 78.86% | 90.47% | 82.06% | 0.861 | 72.74% | 77.80% | 0.752 | 74.80% | 76.29% | 0.755 |
| | SESSION | 86.30% | 90.15% | 94.70% | 0.924 | 90.19% | 75.97% | 0.825 | 77.87% | 90.18% | 0.836 |
| | Senti4SD | 95.27% | 97.25% | 97.45% | 0.974 | 95.02% | 93.51% | 0.943 | 93.15% | 95.01% | 0.941 |
| | EASTER | **97.04%** | **98.04%** | **98.23%** | **0.981** | **97.47%** | **95.57%** | **0.965** | **95.21%** | **97.59%** | **0.964** |
| Stack Overflow 1500 | SentiStrength | 68.00% | 19.28% | 36.64% | 0.253 | **86.20%** | 74.98% | 0.802 | 36.74% | **44.38%** | 0.402 |
| | SentiStrength-SE | 78.00% | 31.18% | 22.14% | 0.259 | 82.72% | 92.86% | 0.875 | 50.00% | 19.66% | 0.282 |
| | SESSION | 78.13% | 30.89% | 29.01% | **0.299** | 85.10% | 89.67% | 0.873 | 54.10% | 37.08% | **0.440** |
| | Senti4SD | 76.93% | 27.59% | **30.53%** | 0.290 | 83.11% | 90.51% | 0.867 | 62.07% | 20.22% | 0.305 |
| | EASTER | **80.05%** | **32.90%** | 19.23% | 0.243 | 82.87% | **94.63%** | **0.884** | **76.19%** | 26.97% | 0.398 |
| App Reviews | SentiStrength | 67.45% | 71.81% | 87.63% | 0.789 | 4.76% | 4.00% | 0.043 | 70.97% | 50.77% | 0.592 |
| | SentiStrength-SE | 61.58% | 74.15% | 81.72% | 0.777 | 9.59% | **28.00%** | 0.143 | 80.95% | 39.23% | 0.528 |
| | SESSION | 68.62% | 76.17% | 87.63% | 0.815 | 9.76% | 16.00% | 0.121 | 77.91% | 51.54% | 0.620 |
| | Senti4SD | 63.93% | 71.24% | 86.56% | 0.782 | 9.80% | 20.00% | 0.132 | 81.25% | 40.00% | 0.536 |
| | EASTER | **75.07%** | **80.47%** | **93.01%** | **0.863** | **14.71%** | 20.00% | **0.169** | **84.78%** | **60.00%** | **0.703** |
| JIRA Issues | SentiStrength | **81.21%** | 86.03% | 93.45% | 0.896 | —— | —— | —— | 98.16% | **75.63%** | **0.854** |
| | SentiStrength-SE | 77.21% | **95.26%** | 90.00% | 0.926 | —— | —— | —— | 99.34% | 71.38% | 0.831 |
| | SESSION | 80.56% | 93.13% | 93.45% | 0.933 | —— | —— | —— | 98.55% | 74.69% | 0.850 |
| | Senti4SD | 57.88% | 81.55% | 86.90% | 0.841 | —— | —— | —— | 99.65% | 44.65% | 0.617 |
| | EASTER | 80.78% | 93.98% | **96.90%** | **0.954** | —— | —— | —— | **99.79%** | 73.43% | 0.846 |

while in the training set of EASTER, the number of each polarity is relatively balanced ). The extreme sentiment distribution of **App Reviews** is actually a challenge for all tools. We found that except for EASTER, the overall accuracies of all tools are lower than 70%. On **JIRA Issues**, which has a similar sentiments distribution with no neutral texts, EASTER's overall accuracy is only 0.43% lower than the highest (81.21%), while the overall accuracy of Senti4SD on this dataset is only 57.88%, which is significantly lower, showing that its generalizability is not satisfying. Thus, *to answer RQ1*, we argue that EASTER can comprehensively outperform the baselines and achieve better generalizability.

### B. *RQ2: How many contributions do the RoBERTa make?*

By comparing the performance of EASTER and the pure TextCNN on the testing set and the benchmark in Table 2 and Table 4, we can find that EASTER is much better than the pure TextCNN. Moreover, the bigger the difference between the sentimental bias of the dataset and the one of the training set, the more obvious the advantage of EASTER is. On the testing set and **Stack Overflow 1500**, the overall accuracies of EASTER are 11.16% and 6.98% higher than those of the pure TextCNN respectively. On **App Reviews** and **JIRA Issues**, whose sentimental bias is completely different from the training set, EASTER's overall accuracies are 23.46% and 36.83% higher than TextCNN's respectively. In Table 4, we found that Pure TextCNN performs poorly on **App Reviews** and **JIRA Issues** and is not as generalizable as EASTER. Thus, *to answer RQ2*, we argue that the performance gap between EASTER and Pure TextCNN is brought by RoBERTa, showing that a well-proposed pre-trained model with good fine-tuning effectively contributes to EASTER.

### C. *RQ3: How many contributions do the TextCNN make?*

By comparing the performance of EASTER and the pure RoBERTa on the testing set and the benchmark in Table 2 and Table 4, we can find that EASTER is superior to the pure RoBERTa in all overall accuracies. The smallest improvement that TextCNN can bring appears on the **Stack Overflow 1500**, where the overall accuracy of EASTER is 0.38% higher than that of the pure RoBERTa. The biggest improvement appears on the **App Reviews**, where the overall accuracy of EASTER is 16.13% higher. Thus, *to answer RQ3*, TextCNN effectively uses the embeddings brought by RoBERTa (with residual connections) to improve EASTER's performance.

We then randomly observed some of the analyzed results generated by EASTER and other baselines. We first found that EASTER can identify more terms representing expressed sentiments, including "**+1**" (*"And it fits with MVVM. +1"*) and "**thx**" (*"mechanize ended up giving me the most functionality...thx"*) for positive sentiments, and "**-1**" (*"-1 This isn't language agnostic."*) and "**fart**" (*"Brain fart."*) for negative sentiments, while all baseline approaches mark the above four sentences as neutral. We also found that EASTER can distinguish some inverted sentiments caused by the context of the sentimental words. For example, EASTER can correctly mark "*I figured out the problem.*" as neutral, while the other four baselines mark this sentence as negative due to the word "**problem**". For a more sarcastic sentence "*cool just freezes everytime*" that about complains an APP, EASTER can get rid of the effect of "**cool**" by recognizing the negative sentiment from "**freeze**", while the other four baselines all mark this sentence as positive.

We made two additional observations on the experiment results. First, the overall accuracy of Senti4SD is slightly different from the previously reported results[5]. We found that it is because we do not directly use the same training set and testing set as those in the work of Senti4SD. Instead, we follow the same way of dividing training set (70%) and the testing set (30%) randomly, and we use the same the testing set to evaluate both EASTER and Senti4SD. Furthermore, the overall accuracy of Senti4SD in our testing set is slightly higher than the one reported in its work. Thus, we suggest that we make no bias in comparing EASTER and Senti4SD. Second, when evaluating EASTER (trained under Stack Overflow 4423) on the other three datasets (i.e., Stack Overflow

**Table 4: The performances of EASTER, Pure RoBERTa, and Pure TextCNN (all three of them are trained on the same training set containing 70% of Stack Overflow 4423) on the four datasets**

| Dataset | Tool | Overall Accuracy | Positive | | | Neutral | | | Negative | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | P | R | F | P | R | F | P | R | F |
| Stack Overflow 4423 | EASTER | **97.04%** | **98.04%** | **98.23%** | **0.981** | **97.47%** | **95.57%** | **0.965** | **95.21%** | **97.59%** | **0.964** |
| | Pure RoBERTa | 93.60% | **98.04%** | 91.88% | 0.949 | 89.36% | 95.22% | 0.922 | 94.69% | 93.51% | 0.941 |
| | Pure TextCNN | 87.84% | 93.62% | 91.23% | 0.924 | 88.35% | 83.71% | 0.860 | 80.75% | 89.35% | 0.848 |
| Stack Overflow 1500 | EASTER | **80.05%** | **32.90%** | **19.23%** | **0.243** | **82.87%** | 94.63% | **0.884** | 76.19% | 26.97% | **0.398** |
| | Pure RoBERTa | 79.67% | 21.74% | 7.63% | 0.113 | 81.67% | **96.14%** | 0.883 | **76.92%** | 22.47% | 0.348 |
| | Pure TextCNN | 73.07% | 27.54% | 14.50% | 0.190 | 82.62% | 85.39% | 0.840 | 30.00% | **33.71%** | 0.317 |
| App Reviews | EASTER | **75.07%** | **80.47%** | **93.01%** | **0.863** | **14.71%** | 20.00% | 0.169 | 84.78% | **60.00%** | **0.703** |
| | Pure RoBERTa | 58.94% | 78.98% | 74.73% | 0.768 | 11.93% | **52.00%** | **0.194** | **87.50%** | 37.69% | 0.527 |
| | Pure TextCNN | 51.61% | 72.12% | 63.98% | 0.678 | 10.20% | 40.00% | 0.163 | 60.26% | 36.15% | 0.452 |
| JIRA Issues | EASTER | **80.78%** | **93.98%** | **96.90%** | **0.954** | —— | —— | —— | 99.79% | **73.43%** | **0.846** |
| | Pure RoBERTa | 67.93% | 90.88% | 82.41% | **0.864** | —— | —— | —— | **100.00%** | 61.32% | 0.760 |
| | Pure TextCNN | 43.95% | 85.47% | 68.97% | 0.763 | —— | —— | —— | 89.61% | 32.55% | 0.478 |

1500, App Reviews, and Jira Issues), the overall accuracy falls to around 80% (from 75% to 81%) on all three datasets. This result is actually quite surprising to us because we expect that: (1) the trained EASTER performs better on **Stack Overflow 1500**, which has the similar percentage of three kinds of sentiments with **Stack Overflow 4423**; (2) the trained EASTER performs worse on **App Reviews**, and **Jira Issues**, which almost contains only positive and negative sentiments. In fact, the overall accuracy of trained EASTER on **Jira Issues** (with no neutral sentiments) is even 0.73% higher than the one on **Stack Overflow 1500**. This observation indicates that: (1) the generalizability of EASTER is enhanced by incorporating RoBERTa into TextCNN, especially when compared to the performance of Senti4SD on different datasets; (2) how human markers label the sentiments on SE texts plays a more important role for the performance of automated sentiment analysis on SE texts, instead of either the percentage of different sentiments in the set of SE texts, or the software information sites where the SE texts are collected. We will investigate why and how different human markers label the sentiments on SE texts quite differently in future.

## 6 THREATS TO VALIDITY

One possible threat is that the pre-trained model of RoBERTa used in this work is re-trained and fine-tuned under the training set of 58M tweets with labeled sentiments, instead of sentimentally-labeled, SE-specified texts. The work of Senti4SD claims that their word embedding is based on a DSM that is particularly built upon 20M texts from Stack Overflow and the authors provide the trained DSM online. However, in their provided DSM, words are represented as high-dimensional vectors. This situation makes the DSM very sparse and thus not suitable to cooperate with TextCNN. Unfortunately, we also cannot find the 20M Stack Overflow texts to re-train RoBERTa in the SE domain. However, the evaluation shows that our incorporated version of RoBERTa is able to help EASTER achieve the overall better accuracy and generalizability than baseline approaches in analyzing sentiments on SE texts, though in this paper we do not aim to propose novel architectures in our training framework. Building a large dataset of SE texts to re-train a SE-specified version of RoBERTa is in future work. We also noticed that multiple enhanced, pre-trained models (e.g., CodeBERT [15] and GraphCodeBERT [18]) are proposed to enhance code-related learning tasks. Unfortunately, we found that most of our evaluated SE texts focus more on using natural language to express their judgments and sentiments with a few referred code identifiers, instead of discussing relatively complete code snippets written in programming languages. Furthermore, **App reviews** generally contains no code elements and **Stack Overflow 1500** replaces specific code identifiers with unified "CODE_FRAGMENT". These situations make EASTER difficult to further take advantage of code-related, pre-trained models. In future work, we plan to further explore the cause of sentiments (also a research hot-spot in the field of sentiment analysis, such as [12]) on SE texts by enhancing these texts with the code texts of referred code elements, where we believe that CodeBERT or GraphCodeBERT will play a vital role.

Another threat is that EASTER is only trained under the dataset with 4423 Stack Overflow posts. The reason is that: (1) compared to the other three datasets, **Stack Overflow 4423** is the largest and contains balanced types of labeled sentiments (i.e., positive, neutral, and negative); (2) Senti4SD, which is one of the state-of-the-art learning-based sentiment analysis on SE texts, is only trained and evaluated on **Stack Overflow 4423**, and we want to make a fair comparison with Senti4SD. Additionally, we use the other three datasets to evaluate the generalizability of trained EASTER.

Our evaluation is based on the four datasets containing 7,190 texts with manually labeled datasets in total, the size of which is not large. But we still consider our findings relevant because the four datasets come from two existing works and are created from three different software information sets (i.e., Stack Overflow, app stores, and Jira). The two datasets from Stack Overflow represent developers' typical expressed sentiments in SE texts. The other two datasets contain SE texts that are mainly labeled as either positive or negative sentiments. Thus, these two datasets are very helpful to verify whether EASTER is over-fitted to the neutral sentiment, which is the majority sentiments in the two Stack Overflow datasets, in SE texts. Our evaluation shows that EASTER (trained on **Stack Overflow 4423** only) still performs best on **App Reviews**, and only slightly worse on **JIRA Issues**, where SentiStreng th-SE and Senti4SD (the two SE-customized baseline approaches) suffer a visible loss in their performance.

## 7 CONCLUSIONS AND FUTURE WORK

In this paper, we proposed an integrated training framework for sentiment polarity classification on SE texts through the pre-trained transformer model RoBERTa for word embedding, and the sentence-classification oriented CNN architecture TextCNN for downstream classification tasks. The goal of our proposed training framework is to better capture the unique expression of sentiments in the SE texts. Our evaluation based on four datasets from Stack Overflow, app stores, and Jira shows that our approach can achieve both good accuracy and generalizability when compared with four baseline approaches. Our replication package is now publicly available [13].

Our future works are as follows: (1) we plan to further improve the generalizability of our learning-based approach, i.e., the accuracy of the trained model when applied to different sets of SE texts, to better support software development in practice; (2) we plan to use machine-learning techniques to enhance the depth of manual exploration for better understanding of the unique expression of sentiments in SE texts by consulting existing work (e.g., [8]).

## ACKNOWLEDGEMENT

Xuefei Tao also contributes to the implementation part of this study. This work is jointly supported by the National Key Research and Development Program of China (No.2019YFE0105500) and the Research Council of Norway (No.309494), as well as the National Natural Science Foundation of China (Grants No.62072227, 61802173, and 61690204), Intergovernmental Bilateral Innovation Project of Jiangsu Province (BZ2020017), and the Collaborative Innovation Center of Novel Software Technology and Industrialization.

## REFERENCES

[1] Toufique Ahmed, Amiangshu Bosu, Anindya Iqbal, and Shahram Rahimi. 2017. SentiCR: a customized sentiment analysis tool for code review interactions. In *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering, ASE 2017, Urbana, IL, USA, October 30 - November 03, 2017*, Grigore Rosu, Massimiliano Di Penta, and Tien N. Nguyen (Eds.). IEEE Computer Society, 106–111. https://doi.org/10.1109/ASE.2017.8115623

[2] Francesco Barbieri, Jose Camacho-Collados, Leonardo Neves, and Luis Espinosa-Anke. 2020. Tweeteval: Unified benchmark and comparative evaluation for tweet classification. *arXiv preprint arXiv:2010.12421* (2020).

[3] Andrew Begel, Robert DeLine, and Thomas Zimmermann. 2010. Social media for software engineering. In *Proceedings of the Workshop on Future of Software Engineering Research, FoSER 2010, at the 18th ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2010, Santa Fe, NM, USA, November 7-11, 2010*, Gruia-Catalin Roman and Kevin J. Sullivan (Eds.). ACM, 33–38. https://doi.org/10.1145/1882362.1882370

[4] Joel Brandt, Philip J. Guo, Joel Lewenstein, Mira Dontcheva, and Scott R. Klemmer. 2009. Two studies of opportunistic programming: interleaving web foraging, learning, and writing code. In *Proceedings of the 27th International Conference on Human Factors in Computing Systems, CHI 2009, Boston, MA, USA, April 4-9, 2009*, Dan R. Olsen Jr., Richard B. Arthur, Ken Hinckley, Meredith Ringel Morris, Scott E. Hudson, and Saul Greenberg (Eds.). ACM, 1589–1598. https://doi.org/10.1145/1518701.1518944

[5] Fabio Calefato, Filippo Lanubile, Federico Maiorano, and Nicole Novielli. 2018. Sentiment polarity detection for software development. *Empirical Software Engineering* 23, 3 (2018), 1352–1382.

[6] Fabio Calefato, Filippo Lanubile, and Nicole Novielli. 2018. How to ask for technical help? Evidence-based guidelines for writing questions on Stack Overflow. *Information and Software Technology* 94 (2018), 186–207.

[7] Giovanni Capobianco, Andrea De Lucia, Rocco Oliveto, Annibale Panichella, and Sebastiano Panichella. 2013. Improving IR-based traceability recovery via noun-based indexing of software artifacts. *J. Softw. Evol. Process.* 25, 7 (2013), 743–762. https://doi.org/10.1002/smr.1564

[8] Alex Davies, András Juhász, Marc Lackenby, and Nenad Tomasev. 2021. The signature and cusp geometry of hyperbolic knots. *CoRR* abs/2111.15323 (2021). arXiv:2111.15323 https://arxiv.org/abs/2111.15323

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* abs/1810.04805 (2018). arXiv:1810.04805 http://arxiv.org/abs/1810.04805

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[11] Jin Ding, Hailong Sun, Xu Wang, and Xudong Liu. 2018. Entity-level sentiment analysis of issue comments. In *Proceedings of the 3rd International Workshop on Emotion Awareness in Software Engineering, SEmotion@ICSE 2018, Gothenburg, Sweden, June 2, 2018*, Andrew Begel, Alexander Serebrenik, and Daniel Graziotin (Eds.). ACM, 7–13. https://doi.org/10.1145/3194932.3194935

[12] Zixiang Ding, Rui Xia, and Jianfei Yu. 2020. ECPE-2D: Emotion-Cause Pair Extraction based on Joint Two-Dimensional Representation, Interaction and Prediction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault (Eds.). Association for Computational Linguistics, 3161–3170. https://doi.org/10.18653/v1/2020.acl-main.288

[13] EASTER. 2022. Incorporating Pre-trained Transformer Models into TextCNN for Sentiment Analysis on Software Engineering Texts. https://github.com/xiaobo-lab/EASTER

[14] Dumitru Erhan, Yoshua Bengio, Aaron C. Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010. Why Does Unsupervised Pre-training Help Deep Learning? *J. Mach. Learn. Res.* 11 (2010), 625–660. https://dl.acm.org/citation.cfm?id=1756025

[15] Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, and Ming Zhou. 2020. CodeBERT: A Pre-Trained Model for Programming and Natural Languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020 (Findings of ACL, Vol. EMNLP 2020)*, Trevor Cohn, Yulan He, and Yang Liu (Eds.). Association for Computational Linguistics, 1536–1547. https://doi.org/10.18653/v1/2020.findings-emnlp.139

[16] Gianmarco Fucci, Nathan Cassee, Fiorella Zampetti, Nicole Novielli, Alexander Serebrenik, and Massimiliano Di Penta. 2021. Waiting around or job half-done? Sentiment in self-admitted technical debt. In *18th IEEE/ACM International Conference on Mining Software Repositories, MSR 2021, Madrid, Spain, May 17-19, 2021*. IEEE, 403–414. https://doi.org/10.1109/MSR52588.2021.00052

[17] Latifa Guerrouj, Shams Azad, and Peter C. Rigby. 2015. The influence of App churn on App success and StackOverflow discussions. In *22nd IEEE International Conference on Software Analysis, Evolution, and Reengineering, SANER 2015, Montreal, QC, Canada, March 2-6, 2015*, Yann-Gaël Guéhéneuc, Bram Adams, and Alexander Serebrenik (Eds.). IEEE Computer Society, 321–330. https://doi.org/10.1109/SANER.2015.7081842

[18] Daya Guo, Shuo Ren, Shuai Lu, Zhangyin Feng, Duyu Tang, Shujie Liu, Long Zhou, Nan Duan, Alexey Svyatkovskiy, Shengyu Fu, Michele Tufano, Shao Kun Deng, Colin B. Clement, Dawn Drain, Neel Sundaresan, Jian Yin, Daxin Jiang, and Ming Zhou. 2021. GraphCodeBERT: Pre-training Code Representations with Data Flow. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net. https://openreview.net/forum?id=jLoC4ez43PZ

[19] Emitza Guzman, David Azócar, and Yang Li. 2014. Sentiment analysis of commit comments in GitHub: an empirical study. In *11th Working Conference on Mining Software Repositories, MSR 2014, Proceedings, May 31 - June 1, 2014, Hyderabad, India*, Premkumar T. Devanbu, Sung Kim, and Martin Pinzger (Eds.). ACM, 352–355. https://doi.org/10.1145/2597073.2597118

[20] Emitza Guzman and Walid Maalej. 2014. How Do Users Like This Feature? A Fine Grained Sentiment Analysis of App Reviews. In *IEEE 22nd International Requirements Engineering Conference, RE 2014, Karlskrona, Sweden, August 25-29, 2014*, Tony Gorschek and Robyn R. Lutz (Eds.). IEEE Computer Society, 153–162. https://doi.org/10.1109/RE.2014.6912257

[21] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580* (2012).

[22] Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146* (2018).

[23] Zijie Huang, Zhiqing Shao, Guisheng Fan, Jianhua Gao, Ziyi Zhou, Kang Yang, and Xingguang Yang. 2021. Predicting Community Smells' Occurrence on Individual Developers by Sentiments. In *29th IEEE/ACM International Conference on Program Comprehension, ICPC 2021, Madrid, Spain, May 20-21, 2021*. IEEE, 230–241. https://doi.org/10.1109/ICPC52881.2021.00030

[24] Md Rakibul Islam and Minhaz F Zibran. 2018. SentiStrength-SE: Exploiting domain specificity for improved sentiment analysis in software engineering text. *Journal of Systems and Software* 145 (2018), 125–146.

[25] Robbert Jongeling, Proshanta Sarkar, Subhajit Datta, and Alexander Serebrenik. 2017. On negative results when using sentiment analysis tools for software engineering research. *Empir. Softw. Eng.* 22, 5 (2017), 2543–2584. https://doi.org/10.1007/s10664-016-9493-x

[26] Y. Kim. 2014. Convolutional Neural Networks for Sentence Classification. *Eprint Arxiv* (2014).

[27] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[28] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. 2015. Deeply-supervised nets. In *Artificial intelligence and statistics*. PMLR, 562–570.

[29] Bin Lin, Fiorella Zampetti, Gabriele Bavota, Massimiliano Di Penta, Michele Lanza, and Rocco Oliveto. 2018. Sentiment analysis for software engineering: How far can we go?. In *Proceedings of the 40th international conference on software engineering*. 94–104.

[30] Jinfeng Lin, Yalin Liu, Qingkai Zeng, Meng Jiang, and Jane Cleland-Huang. 2021. Traceability Transformed: Generating more Accurate Links with Pre-Trained BERT Models. In *43rd IEEE/ACM International Conference on Software Engineering, ICSE 2021, Madrid, Spain, 22-30 May 2021*. IEEE, 324–335. https://doi.org/10.1109/ICSE43902.2021.00040

[31] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR* abs/1907.11692 (2019). arXiv:1907.11692 http://arxiv.org/abs/1907.11692

[32] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).

[33] Mika Mäntylä, Bram Adams, Giuseppe Destefanis, Daniel Graziotin, and Marco Ortu. 2016. Mining valence, arousal, and dominance: possibilities for detecting burnout and productivity?. In *Proceedings of the 13th International Conference on Mining Software Repositories, MSR 2016, Austin, TX, USA, May 14-22, 2016*, Miryung Kim, Romain Robbes, and Christian Bird (Eds.). ACM, 247–258. https://doi.org/10.1145/2901739.2901752

[34] Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in Translation: Contextualized Word Vectors. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 6294–6305. https://proceedings.neurips.cc/paper/2017/hash/20c86a628232a67e7bd46f76fba7ce12-Abstract.html

[35] Daniel J. McDuff, Amy K. Karlson, Ashish Kapoor, Asta Roseway, and Mary Czerwinski. 2012. AffectAura: an intelligent system for emotional memory. In *CHI Conference on Human Factors in Computing Systems, CHI '12, Austin, TX, USA - May 05 - 10, 2012*, Joseph A. Konstan, Ed H. Chi, and Kristina Höök (Eds.). ACM, 849–858. https://doi.org/10.1145/2207676.2208525

[36] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1301.3781

[37] Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger (Eds.). 3111–3119. https://proceedings.neurips.cc/paper/2013/hash/9aa42b31882ec039965f3c4923ce901b-Abstract.html

[38] Nicole Novielli, Fabio Calefato, and Filippo Lanubile. 2015. The challenges of sentiment detection in the social programmer ecosystem. In *Proceedings of the 7th International Workshop on Social Software Engineering, SSE 2015, Bergamo, Italy, September 1, 2015*, Imed Hammouda and Alberto Sillitti (Eds.). ACM, 33–40. https://doi.org/10.1145/2804381.2804387

[39] Marco Ortu, Bram Adams, Giuseppe Destefanis, Parastou Tourani, Michele Marchesi, and Roberto Tonelli. 2015. Are Bullies More Productive? Empirical Study of Affectiveness vs. Issue Fixing Time. In *12th IEEE/ACM Working Conference on Mining Software Repositories, MSR 2015, Florence, Italy, May 16-17, 2015*, Massimiliano Di Penta, Martin Pinzger, and Romain Robbes (Eds.). IEEE Computer Society, 303–313. https://doi.org/10.1109/MSR.2015.35

[40] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, Alessandro Moschitti, Bo Pang, and Walter Daelemans (Eds.). ACL, 1532–1543. https://doi.org/10.3115/v1/d14-1162

[41] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, Marilyn A. Walker, Heng Ji, and Amanda Stent (Eds.). Association for Computational Linguistics, 2227–2237. https://doi.org/10.18653/v1/n18-1202

[42] Daniel Pletea, Bogdan Vasilescu, and Alexander Serebrenik. 2014. Security and emotion: sentiment analysis of security discussions on GitHub. In *11th Working Conference on Mining Software Repositories, MSR 2014, Proceedings, May 31 - June 1, 2014, Hyderabad, India*, Premkumar T. Devanbu, Sung Kim, and Martin Pinzger (Eds.). ACM, 348–351. https://doi.org/10.1145/2597073.2597117

[43] Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained Models for Natural Language Processing: A Survey. *CoRR* abs/2003.08271 (2020). arXiv:2003.08271 https://arxiv.org/abs/2003.08271

[44] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding with unsupervised learning. (2018).

[45] Mohammad Masudur Rahman, Chanchal K. Roy, and Iman Keivanloo. 2015. Recommending insightful comments for source code using crowdsourced knowledge. In *15th IEEE International Working Conference on Source Code Analysis and Manipulation, SCAM 2015, Bremen, Germany, September 27-28, 2015*, Michael W. Godfrey, David Lo, and Foutse Khomh (Eds.). IEEE Computer Society, 81–90. https://doi.org/10.1109/SCAM.2015.7335404

[46] Adrian Schröter, Jorge Aranda, Daniela E. Damian, and Irwin Kwan. 2012. To talk or not to talk: factors that influence communication around changesets. In *CSCW '12 Computer Supported Cooperative Work, Seattle, WA, USA, February 11-15, 2012*, Steven E. Poltrock, Carla Simone, Jonathan Grudin, Gloria Mark, and John Riedl (Eds.). ACM, 1317–1326. https://doi.org/10.1145/2145204.2145401

[47] Vinayak Sinha, Alina Lazar, and Bonita Sharif. 2016. Analyzing developer sentiment in commit logs. In *Proceedings of the 13th International Conference on Mining Software Repositories, MSR 2016, Austin, TX, USA, May 14-22, 2016*, Miryung Kim, Romain Robbes, and Christian Bird (Eds.). ACM, 520–523. https://doi.org/10.1145/2901739.2903501

[48] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*. ACL, 1631–1642. https://aclanthology.org/D13-1170/

[49] Agus Sulistya, Gede Artha Azriadi Prana, Abhishek Sharma, David Lo, and Christoph Treude. 2020. SIEVE: Helping developers sift wheat from chaff via cross-platform analysis. *Empir. Softw. Eng.* 25, 1 (2020), 996–1030. https://doi.org/10.1007/s10664-019-09775-w

[50] Kexin Sun, Hui Gao, Hongyu Kuang, Xiaoxing Ma, Guoping Rong, Dong Shao, and He Zhang. 2021. Exploiting the Unique Expression for Improved Sentiment Analysis in Software Engineering Text. In *29th IEEE/ACM International Conference on Program Comprehension, ICPC 2021, Madrid, Spain, May 20-21, 2021*. IEEE, 149–159. https://doi.org/10.1109/ICPC52881.2021.00023

[51] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1–9.

[52] Mike Thelwall, Kevan Buckley, and Georgios Paltoglou. 2012. Sentiment strength detection for the social web. *Journal of the American Society for Information Science and Technology* 63, 1 (2012), 163–173.

[53] Bogdan Vasilescu, Vladimir Filkov, and Alexander Serebrenik. 2013. StackOverflow and GitHub: Associations between Software Development and Crowdsourced Knowledge. In *International Conference on Social Computing, SocialCom 2013, SocialCom/PASSAT/BigData/EconCom/BioMedCom 2013, Washington, DC, USA, 8-14 September, 2013*. IEEE Computer Society, 188–195. https://doi.org/10.1109/SocialCom.2013.35

[54] Liang Wang, Tao Gu, Alex X. Liu, Hengzhi Yao, Xianping Tao, and Jian Lu. 2019. Assessing User Mental Workload for Smartphone Applications With Built-In Sensors. *IEEE Pervasive Comput.* 18, 1 (2019), 59–70. https://doi.org/10.1109/MPRV.2018.2873851

[55] Westley Weimer. 2019. What goes on in your brain when you read and understand code?. In *Proceedings of the 27th International Conference on Program Comprehension, ICPC 2019, Montreal, QC, Canada, May 25-31, 2019*, Yann-Gaël Guéhéneuc, Foutse Khomh, and Federica Sarro (Eds.). IEEE / ACM, 1. https://doi.org/10.1109/ICPC.2019.00013

[56] Michal R. Wróbel. 2013. Emotions in the software development process. In *6th International Conference on Human System Interactions, HSI 2013, Sopot, Poland, June 6-8, 2013*. IEEE, 518–523. https://doi.org/10.1109/HSI.2013.6577875

[57] Ting Zhang, Bowen Xu, Ferdian Thung, Stefanus Agus Haryono, David Lo, and Lingxiao Jiang. 2020. Sentiment Analysis for Software Engineering: How Far Can Pre-trained Transformer Models Go?. In *IEEE International Conference on Software Maintenance and Evolution, ICSME 2020, Adelaide, Australia, September 28 - October 2, 2020*. IEEE, 70–80. https://doi.org/10.1109/ICSME46990.2020.00017

[58] Ting Zhang, Bowen Xu, Ferdian Thung, Stefanus Agus Haryono, David Lo, and Lingxiao Jiang. 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)*. IEEE, 70–80.