Emoji-powered Sentiment and Emotion Detection from Software Developers' Communication Data

ZHENPENG CHEN, YANBIN CAO, and HUIHAN YAO, Key Lab of High-Confidence Software Technology, MoE (Peking University), China
XUAN LU, University of Michigan, USA
XIN PENG, Fudan University, China
HONG MEI and XUANZHE LIU, Key Lab of High-Confidence Software Technology, MoE (Peking University), China

Sentiment and emotion detection from textual communication records of developers have various application scenarios in software engineering (SE). However, commonly used off-the-shelf sentiment/emotion detection tools cannot obtain reliable results in SE tasks and misunderstanding of technical knowledge is demonstrated to be the main reason. Then researchers start to create labeled SE-related datasets manually and customize SE-specific methods. However, the scarce labeled data can cover only very limited lexicon and expressions. In this article, we employ emojis as an instrument to address this problem. Different from manual labels that are provided by annotators, emojis are self-reported labels provided by the authors themselves to intentionally convey affective states and thus are suitable indications of sentiment and emotion in texts. Since emojis have been widely adopted in online communication, a large amount of emoji-labeled texts can be easily accessed to help tackle the scarcity of the manually labeled data. Specifically, we leverage Tweets and GitHub posts containing emojis to learn representations of SE-related texts through emoji prediction. By predicting emojis containing in each text, texts that tend to surround the same emoji are represented with similar vectors, which transfers the sentiment knowledge contained in emoji usage to the representations of texts. Then we leverage the sentiment-aware representations as well as manually labeled data to learn the final sentiment/emotion classifier via transfer learning. Compared to existing approaches, our approach can achieve significant improvement on representative benchmark datasets, with an average increase of 0.036 and 0.049 in macro-F1 in sentiment and emotion detection, respectively. Further investigations reveal that the large-scale Tweets make a key contribution to the power of our approach. This finding informs future research not to unilaterally pursue the domain-specific resource but try to transform knowledge from the open domain through ubiquitous signals such as emojis. Finally, we present the open challenges of sentiment and emotion detection in SE through a qualitative analysis of texts misclassified by our approach.

This work was supported by the National Key R&D Program of China under the grant number 2018YFB1004800, the Beijing Outstanding Young Scientist Program under the grant number BJJWZYJH01201910001004, the National Natural Science Foundation of China under grant numbers J1924032 and 61725201, and the Key Laboratory of Intelligent Passenger Service of Civil Aviation.
Authors' addresses: Z. Chen, Y. Cao, H. Yao, H. Mei, and X. Liu (corresponding author), Key Lab of High-Confidence Software Technology, MoE (Peking University), Beijing, China; emails: {czp, caoyanbin, yaohuihan, meih, xzl}@pku.edu.cn; X. Lu, University of Michigan, Ann Arbor, MI; email: luxuan@umich.edu; X. Peng, Fudan University, Shanghai, China; email: pengxin@fudan.edu.cn.

1049-331X/2020/12-ART18 $15.00
https://doi.org/10.1145/3424308

CCS Concepts: • **Information systems → Sentiment analysis**; • **Software and its engineering → Collaboration in software development**;

Additional Key Words and Phrases: Emoji, sentiment, emotion, software engineering

ACM Reference format:
Zhenpeng Chen, Yanbin Cao, Huihan Yao, Xuan Lu, Xin Peng, Hong Mei, and Xuanzhe Liu. 2020. Emoji-powered Sentiment and Emotion Detection from Software Developers' Communication Data. *ACM Trans. Softw. Eng. Methodol.* 30, 2, Article 18 (December 2020), 48 pages.
https://doi.org/10.1145/3424308

1 INTRODUCTION

Software development is a highly collaborative activity that is susceptible to the affective states of developers [40, 55, 75, 81]. Negative affective states may make developers underperform in software projects [33, 81], while positive ones are demonstrated to be correlated with increased productivity [37]. Therefore, awareness of developers' affective states (including sentiment and emotion) is crucial for stakeholders involved in the software development lifecycle. Sentiment is usually considered to have three polarities, i.e., positive, negative, and neutral [62], while emotion refers to love, sadness, anger, joy, surprise, fear, and so on [92]. Many approaches, such as surveys [58], biometric measurements [35], and text analysis [12], have been developed for detecting sentiment or emotion of developers.

Among these approaches, text-based detection has been increasingly popular [61, 104], due to its convenience, low cost, and technical readiness. Many off-the-shelf sentiment/emotion detection tools not designed for SE-related texts have been applied in SE studies, but a recent study has indicated that these tools cannot produce reliable results in some SE tasks [54]. Furthermore, Islam and Zibran [48] applied an off-the-shelf tool SentiStrength [100] to an SE-related dataset and found that misunderstanding of domain-specific meanings of words (namely *technical knowledge* in the rest of this article) accounts for the most misclassifications. Such a finding has inspired a series of research efforts in recent years to create SE-related datasets and develop customized sentiment and emotion detection methods [1, 10, 12, 50, 61].

However, since these customized methods are mainly trained on scarce labeled data (only thousands of samples), they inevitably lack the knowledge of other lexicon and expressions that are not contained in them. Given the large volume of English vocabulary, these missing expressions are indeed non-trivial. To tackle this problem, a straightforward solution is to annotate abundant texts with sentiment/emotion labels. However, manual annotation on a large scale is quite time-consuming and error prone. Instead, recent work in natural language processing (NLP) employed emojis as indications of sentiment/emotion in texts [30]. Different from the manual labels that are provided by annotators based on their perception, emojis are self-reported labels provided by the authors themselves to intentionally convey affective states and thus are suitable indications of sentiment and emotion in written communication. As emojis have become an emerging ubiquitous language used worldwide [65], a large amount of texts containing emojis in social media can be easily accessed to tackle the scarcity of manually labeled texts [30]. Inspired by the previous work [30], we aim to incorporate emoji usage data into sentiment and emotion detection in SE.

In fact, emojis not only pervasively exist in social media [64] but also are adopted in developers' communication to express sentiment [46, 66]. For example, in the GitHub post "thanks for writing this great plugin,👍"[1] the emoji "👍" can be considered an indication of positive sentiment.

[1] https://github.com/MikaAK/s3-plugin-webpack/issues/65.

In this study, we not only employ posts with emojis from Twitter (a typical social media platform) but also consider the posts containing emojis from GitHub (a typical software development platform). Both of these posts are used to complement the scarce manually labeled data. Here, the core insight is as follows: *GitHub posts can provide more technical knowledge beyond the limited labeled data, while posts from Twitter can help learn more general sentiment knowledge that is shared in both technical and non-technical communication.*

Specifically, we propose *SEntiMoji*, an emoji-powered transfer learning (refer to Section 2.4) [36] approach for sentiment and emotion detection in SE. First, through an emoji prediction task, vector representations of texts are derived based on modeling how emojis are used alongside texts on Twitter and GitHub. By predicting emojis contained in each text, texts that tend to surround the same emoji are represented similarly. Through such a process, sentiment knowledge contained in emoji usage is transferred to the representations of texts and thus facilitates the sentiment/emotion classification of these texts. Then, these informative representations are used to predict the manual labels and learn the final sentiment/emotion classifier.

To evaluate the performance of SEntiMoji, we compare it against state-of-the-art sentiment and emotion detection methods in SE on representative benchmark datasets. Results show that SEntiMoji can outperform existing sentiment and emotion detection methods with an average increase of 0.036 and 0.049 in macro-F1, respectively. Further investigations reveal that the GitHub posts do not make a key contribution to the power of SEntiMoji. The combination of large-scale Tweets and a small amount of manually labeled data can achieve satisfactory performance on most tasks. Finally, by manually analyzing the samples misclassified by SEntiMoji, we distill seven and eight common error causes for sentiment and emotion detection, respectively. Some causes, such as implicit sentiment/emotion and complex context information, need special attention from researchers.

The main contributions of this article are as follows:

- We propose an emoji-powered transfer learning approach for sentiment and emotion detection in SE, which utilizes Tweets to capture general sentiment knowledge and GitHub posts as well as manually labeled data to incorporate technical knowledge.
- We demonstrate the effectiveness of SEntiMoji on representative benchmark datasets in SE. Results show that SEntiMoji can significantly improve the state-of-the-art performance on almost all the datasets.
- We investigate the underlying reasons behind the good performance of SEntiMoji by rigorous comparative experiments and provide future research with some insightful implications.
- We manually identify the error causes of SEntiMoji on sentiment and emotion detection, and suggest immediate actions and future research directions based on the findings.
- We have released the data, scripts, trained models, and experimental results used in this study on https://github.com/SEntiMoji/SEntiMoji to facilitate replications or other types of future work.

The rest of this article is organized as follows. Section 2 describes the preliminaries of this work. Section 3 presents the workflow of SEntiMoji in detail. Section 4 describes baseline methods, benchmark datasets, evaluation metrics, and other experimental settings used for evaluation. Section 5 answers four research questions based on the evaluation results. Section 6 summarizes the lessons learned from this study and the implications. Section 7 discusses the threats that could affect the validity of this study. Section 8 summarizes the literature related to this work, followed by concluding remarks in Section 9.

Part of the results in this article have been reported in our previous work at ESEC/FSE 2019 [14]. For those who have read the conference version, this extension mainly includes a new application of SEntiMoji to emotion detection (see Section 4, Section 5.2, and Section 5.3). The qualitative analysis for identifying error causes of SEntiMoji is also completely new (see Section 5.4). Moreover, we include a more comprehensive preliminary section (see Section 2) and related work section (see Section 8). More lessons can be learned from the results of this article (see Section 6).

2 PRELIMINARIES

We start by clarifying the "sentiment" and "emotion" used in this study. Then, we briefly introduce some techniques used in our approach, including word embedding, Long Short-Term Memory (LSTM), and transfer learning.

2.1 Sentiment and Emotion

Sentiment and emotion are both terms related to human subjectivity, so they are sometimes used interchangeably in research without sufficient differentiation, which may lead to poor apprehension and confusion [73]. As this study involves both sentiment and emotion, we need to clearly distinguish them.

2.1.1 Definition of Sentiment and Emotion. Sentiment refers to an attitude, thought, or judgement prompted by a feeling [73]. Usually, in NLP community, sentiment is considered to have three polarities, i.e., positive, negative, and neutral [62]. Emotion refers to a conscious mental reaction subjectively experienced as strong feelings [73]. Different from sentiment, emotion is more sophisticated. So far, there has not been one standard theory on categorizing emotions.

2.1.2 Emotion Models. In psychology domain, a number of empirical and analytical theories about emotions have been proposed. A popular example is a tree-structured classification model of emotions, which is described in Shaver framework [92]. The first level of the tree consists of six primary emotions, i.e., love, sadness, anger, joy, surprise, and fear. Such a tree-structured classification model has been widely adopted in SE studies [75, 78, 81, 83]. Another typical emotion model is VAD model [90], which projects emotions into a bi-dimensional space, where the horizontal dimension indicates the emotional polarities (i.e., valence [5]) and the vertical dimension indicates the levels of reactiveness (i.e., arousal [5]). According to this model, emotions can be represented as combinations of different levels of valence and arousal.

In this study, sentiment detection is considered as determining whether a text expresses a positive, negative, or neutral sentiment, while emotion detection is to identify the presence of specific emotion states from texts. Compared to sentiment detection, *emotion detection is more challenging, because texts expressing a specific emotion are much scarcer than those conveying a kind of sentiment.*

2.2 Word Embedding

Sentiment and emotion detection are both typical NLP tasks. In NLP, to eliminate the discrete nature of words, word embedding techniques, such as skip-gram algorithm [72] and GloVe [85], are proposed to encode every single word into a continuous vector space as a high dimensional vector. Through these techniques, words that commonly occur in a similar context are represented as similar vectors, which can capture the semantic relationship among words. In practice, these techniques are usually performed based on a large amount of natural language texts by utilizing co-occurrence statistics of words in the corpus. For example, the skip-gram algorithm scans each sample in the corpus and uses each word that it has scanned as an input to predict words within a certain range before and after this word; GloVe is learned based on a global word-word co-occurrence matrix, which tabulates how frequently words co-occur with one another in a given

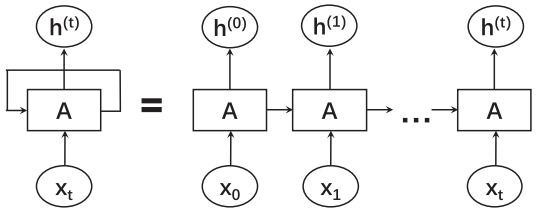

Fig. 1. The structure of RNN.

corpus. Compared to GloVe, the skip-gram algorithm is demonstrated to be more robust and utilize less system resources [59]. In practice, the skip-gram algorithm is widely adopted in SE tasks, such as creating SE-specific word embeddings [28], enhancing software traceability [38], and localizing bugs [114].

2.3 Long Short-Term Memory

Recurrent neural network (RNN) [89] is a kind of neural network specialized for processing sequential data such as texts. It can leverage knowledge from both the past and the current step to predict outcomes. As is illustrated in Figure 1, a typical RNN can be seen as a computational unit (denoted as "A" in the figure) with a self-loop. At each time step t, the unit takes both the current input and its hidden state from the previous time step as the input. Formally, given a sequence of word vectors $[x_1, x_2, \ldots, x_L]$, at time step t, the output $h^{(t)}$ (i.e., the hidden state at time step $t+1$) of the RNN can be computed as:

$$h^{(t)} = f(h^{(t-1)}, x_t). \tag{1}$$

Due to the recurrent nature, RNN is able to capture the sequential information, which is important for NLP tasks. However, due to the well-known gradient vanishing problem, vanilla RNNs are difficult to train to capture long-term dependency for sequential texts. To address this problem, LSTM [44] introduces a gating mechanism to determine when and how the states of hidden layers can be updated. Each LSTM unit contains a memory cell, an input gate, a forget gate, and an output gate. The input gate controls the input activations into the memory cell, and the output gate controls the output flow of cell activations into the rest of the network. The memory cells in LSTM store the sequential states of the network, and each memory cell has a self-loop whose weight is controlled by the forget gate. Formally, given the input $x = [x_1, x_2, \ldots, x_L]$, at time step t, LSTM computes unit states of the network as follows:

$$i^{(t)} = \sigma(U_i x_t + W_i h^{(t-1)} + b_i), \tag{2a}$$

$$f^{(t)} = \sigma(U_f x_t + W_f h^{(t-1)} + b_f), \tag{2b}$$

$$o^{(t)} = \sigma(U_o x_t + W_o h^{(t-1)} + b_o), \tag{2c}$$

$$c^{(t)} = f_t \odot c^{(t-1)} + i^{(t)} \odot tanh(U_c d_t + W_c h^{(t-1)} + b_c), \tag{2d}$$

$$h^{(t)} = o^{(t)} \odot tanh(c^{(t)}), \tag{2e}$$

where $i^{(t)}$, $f^{(t)}$, $o^{(t)}$, $c^{(t)}$, and $h^{(t)}$ denote the states of the input gate, forget gate, output gate, memory cell, and hidden layer at time step t; σ and $tanh$ denote sigmod and tanh activation functions that crop/normalize activation values; W, U, b, and $\odot$ denote the recurrent weights, input weights, biases, and element-wise product, respectively.

In practice, LSTM has been widely adopted in SE studies, such as code search [13], program repair [71], and detection of semantic code clones [107]. For sentiment/emotion detection, LSTM is commonly used with attention mechanism [102], which determines the importance of different

words to the classification result. Such a combination has been demonstrated to be effective in previous studies [105, 113, 117]. In this study, we also employ LSTM with attention, rather than other sophisticated language models (such as ELMo [86] or BERT [23]) that are demonstrated to require exceptionally large computational resources [98].

2.4 Transfer Learning

Transfer learning aims to leverage a large amount of labeled data in a source task to solve a related but different task (namely a target task), even when the source and the target tasks have different distributions of classes [36]. Labeled data are often limited for NLP tasks, especially new ones. For such tasks, training a neural network for a new task from scratch with limited data may result in over-fitting. One approach to getting around this problem is to take a network model, which has been pre-trained based on plenty of labeled data for a related task (i.e., a source task), to perform the target task. This process is so called transfer learning. Assuming the target task is related to the source task, transfer learning enables us to take advantages of the prior efforts on feature extraction (i.e., the pre-trained parameters of the network) in the source task. Due to the effectiveness of transfer learning, it has been employed to tackle a wide range of SE tasks, such as software defect prediction [76], software effort estimation [57], and performance modeling of configurable systems [52].

3 METHODOLOGY

Since sentiment and emotion detection are both relatively new tasks in SE and labeled data for them are not so sufficient, we employ transfer learning to tackle the two problems. Specifically, we use emojis as indications of sentiment/emotion and employ emoji prediction as the source task. On one hand, emojis are able to express various emotions [45, 66]. The rich emotional information contained in emoji usage makes emoji prediction a suitable source task of sentiment and emotion detection. On the other hand, emojis are widely used in social media [64] and developers' commu-nication [46, 66], and thus can be easily collected. The large-scale emoji usage data can complement the scarce manually labeled data for the two target tasks.

We propose SEntiMoji, an emoji-powered transfer learning approach for sentiment and emotion detection in SE. First, we learn sentiment- and emotion-aware representations of texts by using emoji prediction as an instrument. More specifically, we use emojis as indications of sentiment/emotion and learn vector representations of texts by predicting which emojis are used in them. Texts that tend to surround the same emoji are represented as similar vectors. Then these informative representations are used as features to predict the true sentiment/emotion labels. Through these representations, sentiment knowledge contained in emoji usage data is transferred from the emoji prediction task into the sentiment/emotion classifiers.

Since Felbo et al. [30] have released such a representation model (i.e., *DeepMoji*[2]) that is pre-trained based on 56.6 billion Tweets, we directly build SEntiMoji upon the off-the-shelf DeepMoji in a transfer learning way. Specifically, our approach takes two stages: (1) We fine-tune DeepMoji using GitHub posts that contain emojis to incorporate technical knowledge. The fine-tuned model is still a representation model based on emoji prediction, and we call it *DeepMoji-SE*. (2) We use DeepMoji-SE to obtain vector representations of labeled texts, and then use these representations as features to train the sentiment/emotion classifier. We call the final sentiment/emotion classifier *SEntiMoji*. Next, we describe the existing DeepMoji model and the two-stage learning process in detail.

[2]https://github.com/bfelbo/deepmoji.

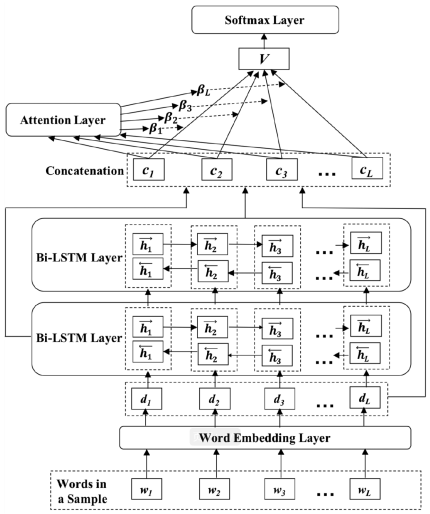

Fig. 2. The architecture of DeepMoji.

3.1 DeepMoji Model

Felbo et al. [30] learned DeepMoji through predicting emojis used in Tweets. To this end, they collected 56.6 billion Tweets (denoted as T), selected the top 64 emojis in this corpus, and excluded the Tweets that do not contain any of these emojis. For each remaining Tweet, they created separate samples for each unique emoji in it. Finally, they balanced the created 1.2 billion samples (denoted as ET) using upsampling and then performed the emoji prediction task.

The model architecture is illustrated in Figure 2. First, for a given sample, words in it are inputted into the word embedding layer that is pre-trained on T. In this step, each word can be represented as a unique vector. Then these word vectors are processed by two bi-directional LSTM layers and one attention layer. Through these steps, the sample can be represented as one vector instead of several word vectors. Finally, the softmax layer treats the vector as the input and outputs the probabilities that this sample may contain each of the 64 emojis. The details of the model architecture are described below.

Word Embedding Layer. The word embedding layer of 256 dimensions is pre-trained based on T with the skip-gram algorithm. A hyperbolic tangent activation function is used to enforce a constraint of each embedding dimension being within $[-1, 1]$. Through this layer, each sample in ET can be denoted as (x, e), where $x = [d_1, d_2, \ldots, d_L]$ denotes the word vector sequences of the plain text removed emoji (d_i as the vector representation of the ith word) and e denotes the emoji contained in the sample.

Bi-Directional LSTM Layer. To take context information (i.e., both past and future words) of the current word at each time step into consideration, DeepMoji employs bi-directional LSTM with 1,024 hidden units (512 in each direction) instead of the traditional LSTM. Each bi-directional LSTM network contains two sub-networks (i.e., a forward network and a backward network) to encode the sequential contexts of each word in the two directions respectively. Given the input $x = [d_1, d_2, \ldots, d_L]$, it computes an encoded vector h_i of each word vector d_i by concatenating the latent vectors from both directions:

$$h_i = \overrightarrow{h_i} \,||\, \overleftarrow{h_i}, \tag{3}$$

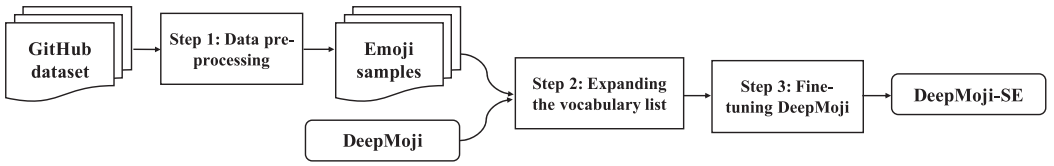

Fig. 3. An overview of the stage 1 of SEntiMoji.

where $\overrightarrow{h_i}$ and $\overleftarrow{h_i}$ denote the forward and backward states of d_i, respectively. To enable the unimpeded information flow in the whole model, the outputs of the two LSTM layers and the word embedding layer are concatenated by the skip-connection algorithm [43], then as input into the attention layer. Specifically, each word of the input sample is further represented as c_i:

$$c_i = d_i||h_{i1}||h_{i2}, \tag{4}$$

where d_i, h_{i1}, and h_{i2} represent the encoded vectors of the ith word extracted from the word embedding layer and the first and second bi-directional LSTM layers.

Attention Layer. Since not all words contribute equally to the overall sentiment/emotion of the sample, the model employs the attention mechanism to determine the importance of each word. The attention score (i.e., the importance) of the ith word is computed as:

$$\alpha_i = \frac{exp(Wc_i)}{\sum_{j=1}^{L} exp(Wc_j)}, \tag{5}$$

where W is the weight matrix of the attention layer. Then the sample can be represented as the weighted sum of all words in it and denoted as V.

Softmax Layer. The final representation V is inputted into the softmax layer to output a 64-dimension probability vector, each element of which denotes the probability that this sample contains one specific emoji.

Taking the real emoji contained in each sample in ET as ground truth, DeepMoji learns parameters by minimizing the cross entropy between the output probability vectors and the one-hot representations of the emojis actually contained in the training samples. Through such a learning process, plain texts that surround the same emoji can be represented similarly.

3.2 Stage 1: Fine-tuning DeepMoji Using GitHub Data

As we use emoji prediction as the source task for SE-customized sentiment and emotion detection, we expect that emoji prediction is also performed in SE contexts. However, the off-the-shelf DeepMoji is trained only on Tweets. Therefore, we need to collect developer-generated texts to incorporate technical knowledge. To this end, we use the conversation data (i.e., issues, issue comments, pull requests, and pull request comments) in the *GitHub dataset* collected by Lu et al. [66], which covers more than one hundred million posts on GitHub, to fine-tune the parameters of DeepMoji. Figure 3 shows an overview of the steps in stage 1.

Step 1: Data pre-processing. We first perform the following procedures to pre-process the posts in the GitHub dataset. We use NLTK[3] to tokenize all the posts into words and convert all the words into lowercase. We remove special symbols, punctuation marks, and non-alphabetical characters to reduce noise. In informal communication, emoticons like ":-)" are frequently used to express sentiment or emotion. To minimize the loss of sentiment information during pre-processing, we use the list of emoticons provided by previous work [42] to identify emoticons and keep them.

[3]https://www.nltk.org/.

Table 1. The Number of Emojis Samples from Twitter and GitHub Associated with Each Emoji in Millions

Emoji	😂	❤	😍	😭	😩	😊	😘	😒	👌	💕	❤	😉	😏	😄	😔	🙌
Twitter	233.7	82.2	79.5	78.1	60.8	54.7	54.6	51.7	50.5	44.0	39.5	39.1	34.8	34.4	32.1	28.1
GitHub	0.013	0.021	0.006	0.010	0.002	0.021	0.002	0.004	0.017	0.001	0.012	0.009	0.006	0.013	0.002	0.005
Emoji	🙏	😳	💯	👍	✌	😉	🙈	😌	😏	😉	😎	🙅	🎶	💗	💜	😅
Twitter	24.8	23.4	21.6	21.0	20.5	20.3	19.9	19.6	18.9	17.5	17.0	16.9	16.1	15.3	15.2	15.0
GitHub	0.011	0.004	0.064	0.578	0.009	0.025	0.006	0.001	0.002	0.093	0.012	0.001	0.002	0.004	0.005	0.018
Emoji	👀	💙	😑	😊	😠	😫	🙊	😕	😀	😜	✋	😞	😡	😐	✨	💀
Twitter	14.9	14.3	14.2	14.2	12.9	12.4	12.0	12.0	11.7	11.7	11.3	11.2	11.1	11.0	11.0	10.8
GitHub	0.013	0.009	0.001	0.003	0.025	0.001	0.002	0.020	0.294	0.010	0.001	0.027	0.002	0.002	0.028	0.002
Emoji	😈	👊	😪	💪	🦵	👐	💔	💗	😁	😖	😓	😠	💁	😊	🎧	😬
Twitter	10.2	9.6	9.5	9.3	9.2	8.9	8.7	8.6	8.1	6.3	6.0	5.7	5.6	5.5	5.4	5.1
GitHub	0.004	0.004	0.001	0.004	0.002	0.001	0.005	0.001	0.001	0.001	0.005	0.002	0.001	0.007	0.001	0.004

Note: Since the emoji samples used for DeepMoji (i.e., those from Twitter) have not been released, the number of them is directly obtained from Felbo et al. [30].

Then we identify URLs, email addresses, references, code snippets, numbers, and mentions from GitHub posts though the Markdown parser[4] and replace them with specific tokens, in case that the concrete contents of them influence the classification of sentiment and emotion. For example, we replace any code snippet with "[code]." In addition, we use regular expressions to identify words with 2 or more repeated characters and replace them with their basic forms. From the pre-processed GitHub data, we extract 809,918 posts containing emojis and then create separate samples for each unique emoji in each post. Since we need to fine-tune DeepMoji, we then select only the samples containing any of the 64 emojis predicted by DeepMoji. Finally, we have 1,058,413 emoji samples remained. For each of the 64 selected emojis, we present the number of corresponding samples extracted from Twitter (i.e., those used for pre-training DeepMoji) and GitHub (i.e., those generated in this step) in Table 1.

Step 2: Expanding the vocabulary list. As many SE-related words are rarely used in Tweets, they are not included in the vocabulary of DeepMoji. As a result, these words would be treated as unknown words by DeepMoji, and technical knowledge contained in them would be lost in the final representation model. To tackle this problem, we extend the initial vocabulary list of DeepMoji. From the emoji samples created in stage 1, we identify a total of 243,986 words that are not included in the word embeddings of DeepMoji as the out-of-vocabulary (OOV) words. Adding all the OOV words into the vocabulary list is neither economical, as it limits model applicability on computation- or memory-constrained scenarios, nor necessary, as many words may contribute little to the end task [29, 115]. Instead, we sort the OOV words in emoji samples by their frequency in a descending order and add the most frequent 3,000 ones, whose occurrences account for 68.89% of those of all the OOV words, into the vocabulary list. We observe that each of other OOV words appear no more than 15 times. As a result, there lacks enough context information to learn their semantics and embeddings, and they may not play an important role in the end task [103]. Therefore, we do not take them into consideration. Finally, the representations of top 3,000 OOV words are randomly initialized in the word embedding layer and then tuned during the fine-tuning process described below.

Step 3: Fine-tuning DeepMoji. In this step, we use the emoji samples extracted in step 1 (denoted as "*EG*") to fine-tune DeepMoji through the chain-thaw approach implemented by Felbo et al. [30]. At a time, the chain-thaw approach fine-tunes a single layer of the network while keeping other layers freezed, which is demonstrated to be able to reduce the risk of over-fitting [30]. Each time we fine-tune, we use plain texts in *EG* as the input and actual emojis in *EG* as the ground truth to minimize the output error of the model. Following the procedures of chain-thaw approach, we first fine-tune the softmax layer and then fine-tune each layer individually starting from the

[4]https://python-markdown.github.io/extensions/api/.

first layer (i.e., the word embedding layer) of the network. Finally, we fine-tune the entire model with all layers unfreezed. We refer to the fine-tuned DeepMoji as *DeepMoji-SE*.

3.3 Stage 2: Training the Sentiment/Emotion Classifiers Using Labeled Data

Based on DeepMoji-SE, we can learn vector representations for SE-related texts. As emotional information contained in emoji usage is transferred into these representations through DeepMoji-SE, the representations are sentiment- and emotion-aware, which can facilitate the sentiment and emotion detection tasks. Since DeepMoji is demonstrated to work for both sentiment and emotion detection, we expect that the fine-tuned DeepMoji-SE can also tackle the two tasks in SE scenarios.

Considering a sentiment or emotion detection task that involves n classes, where n is the number of sentiment polarities or emotions, the training phase is a process of fine-tuning. With other settings of the whole DeepMoji-SE unchanged, we replace its 64-dimension softmax layer with an n-dimension softmax layer. Then, we use the manually labeled data to fine-tune the parameters of the adjusted model. Specifically, we perform the pre-processing procedures described in the *Stage 1* on the labeled SE-related samples and then use the processed samples as input to predict the true sentiment/emotion label. The learning process is performed using the aforementioned chain-thaw approach to minimize the output error of the softmax layer. Finally, we can obtain the sentiment/emotion classifier, which we refer to as SEntiMoji.

4 EVALUATION

We evaluate the performance of SEntiMoji through a benchmark study, where SEntiMoji and baseline methods tackle the same datasets for comparison. The evaluation can be considered as a partial replication of the benchmark study performed by Novielli et al. [79].

4.1 Baseline Methods

So far, several sentiment and emotion detection methods have been used or proposed for SE-related texts. We use them as well as some variants of SEntiMoji as baseline methods.

4.1.1 Existing Sentiment Detection Methods. We employ four existing sentiment detection methods for comparison, including *SentiStrength*, *SentiStrength-SE*, *SentiCR*, and *Senti4SD*. Although SentiStrength is not designed for technical texts like the other three methods, we still take it into consideration as it has been the most popular sentiment detection tool in previous SE studies [48]. We describe these methods briefly:

SentiStrength[5] [100] is a lexicon-based sentiment classifier for informal English texts rather than technical texts. It utilizes a dictionary of several word and phrase lists to compute the sentiment of texts. For an input text, SentiStrength outputs a positive score and a negative score based on its coverage of the built-in dictionary. Based on the algebraic sum of the two scores, SentiStrength can report a trinary score, i.e., 1 (positive), 0 (neutral), or −1 (negative).

SentiStrength-SE[6] [48] is an SE-customized lexicon-based tool adapted from SentiStrength. It is developed based on the results obtained by running SentiStrength on Group-1 of the JIRA dataset (an SE dataset that will be described in Section 4.2). More specifically, Islam and Zibran identified the challenges of SentiStrength to detect sentiment in SE texts. Then, they addressed the majority of the identified challenges by customizing SentiStrength to develop the first SE specific sentiment detection tool SentiStrength-SE. For instance, they adapted the raw inherent dictionary of SentiStrength to contain some SE-related terms.

[5] http://sentistrength.wlv.ac.uk/.
[6] http://laser.cs.uno.edu/Projects/Projects.html.

SentiCR[7] [1] is a supervised sentiment detection method originally proposed for code reviews. It computes TF-IDF [3] of bag-of-words as features and uses traditional machine learning algorithms to train the sentiment classifier.

Senti4SD[8] [10] is a supervised sentiment detection method proposed for developer-generated texts. It leverages three kinds of features, including lexicon-based features (based on the sentimental word list of SentiStrength), keyword-based features (such as uni- and bi-grams), and semantic features (based on the word embeddings trained on large-scale posts collected from Stack Overflow). Finally, it uses Support Vector Machine (SVM) [99] to train the sentiment classifier.

4.1.2 Existing Emotion Detection Methods. For emotion detection, we employ four existing methods for comparison, including *DEVA*, *EmoTxt*, *MarValous*, and *ESEM-E*. All of them are specifically proposed for emotion detection in SE. We then describe these methods briefly:

DEVA[9] [50] is an SE-specific lexicon-based emotion detection tool. It is designed for detecting the presence of four emotional states (i.e., excitement, stress, depression, and relaxation) from technical texts. To this end, it contains two arousal dictionaries (i.e., a general-purpose dictionary named ANEW [106] and a domain-specific dictionary named SEA [70]) and a valence dictionary (i.e., the domain-specific dictionary of SentiStrength-SE). In addition, to increase accuracy, it includes all the heuristics implemented in SentiStrength-SE.

EmoTxt[10] [12] is a supervised learning method proposed for recognizing emotions from technical texts. It leverages four kinds of features, including uni- and bi-grams features (using TF-IDF schema), emotion lexicon features (based on WordNet Affect [97]), politeness features (measured by the tool developed by Danescu et al. [20]), and mood features (measured by the tool developed by De Smedt et al. [94]). Finally, it uses SVM to train the emotion classifier.

MarValous [47] is a supervised learning method proposed for detecting the presence of excitement, stress, depression, and relaxation from technical texts. It leverages seven kinds of features, including n-grams, emoticons, interjections, exclamation marks, and so on. Some of features, such as emoticons and interjections, are specifically defined for the four emotions that it focuses on. Finally, it uses traditional machine learning algorithms to train the emotion classifier.

ESEM-E [74] is a supervised learning method proposed for emotion detection in SE. It leverages uni- and bi-grams as features and uses machine learning algorithms to train the emotion classifier.

4.1.3 Variants of SEntiMoji. SEntiMoji is developed based on three kinds of data, i.e., Twitter data used for training DeepMoji, GitHub data used for training DeepMoji-SE, and manually labeled data used for training the final SEntiMoji. To measure the contribution of the three kinds of data to the overall performance of SEntiMoji, we employ the following seven variants as baseline methods.

SEntiMoji-G1 is a variant of SEntiMoji and shares the same network architecture with SEntiMoji. By comparison, SEntiMoji-G1 leverages GitHub data used by SEntiMoji to train DeepMoji-SE from scratch, rather than to fine-tune the pre-trained DeepMoji. Moreover, to make a fair comparison, SEntiMoji-G1 and SEntiMoji have the same selection of the 64 emojis and both employ the upsampling technique to ensure data balance. Then, following the procedures in *Stage 2*, SEntiMoji-G1 uses the new DeepMoji-SE model to train the sentiment/emotion classifier. Compared to SEntiMoji, SEntiMoji-G1 is trained purely based on GitHub data and manually labeled data, without the incorporation of Twitter data.

[7] https://github.com/senticr/SentiCR/.
[8] https://github.com/collab-uniba/Senti4SD.
[9] https://figshare.com/s/277026f0686f7685b79e.
[10] https://github.com/collab-uniba/Emotion_and_Polarity_SO.

In addition, since here we are not using the pre-trained version of DeepMoji, in fact, we do not have any constraint to select the emojis. To better capture the affective expressions expressed in GitHub posts, we introduce a new variant of SEntiMoji namely *SEntiMoji-G2*, which makes uses of the most frequently used 64 emojis in GitHub. Specifically, the only difference between SEntiMoji-G2 and SEntiMoji-G1 lies in the selection of emojis.

SEntiMoji-T is another variant of SEntiMoji. The only difference between SEntiMoji-T and SEntiMoji is that SEntiMoji-T uses DeepMoji rather than DeepMoji-SE to train the final sentiment/emotion classifier. Compared to SEntiMoji, SEntiMoji-T does not use GitHub data and thus can learn technical knowledge only from the labeled data.

T-80%, T-60%, T-40%,* and *T-20% are variants of SEntiMoji-T. They differ from SEntiMoji-T only in the amount of the labeled training data. They randomly select 80%, 60%, 40%, and 20% of the labeled training data used by SEntiMoji-T to train the sentiment/emotion classifier and keep other settings unchanged.

4.2 Benchmark Datasets

We employ representative datasets covering SE-related texts from different platforms as the benchmark. All the selected datasets are specifically created for sentiment and emotion detection in SE and have been released online.

4.2.1 Datasets for Sentiment Detection. We employ five benchmark datasets (i.e., *JIRA dataset*, *Stack Overflow dataset*, *Code Review dataset*, *Java Library dataset*, and *Unified-S dataset*) for sentiment detection. We then describe these datasets briefly:

JIRA dataset[11] [83] originally contained 5,992 samples extracted from the issue comments on JIRA issue tracking system. It was divided by its authors into three subsets, i.e., Group-1, Group-2, and Group-3. Since SentiStrength-SE is developed by analyzing the Group-1 samples, to compare the methods fairly, we exclude Group-1 from this dataset. The remaining Group-2 and Group-3 contain 1,600 issue comments[12] and 4,000 sentences, respectively. These samples are labeled with love, joy, surprise, anger, sadness, fear, or neutral. In accordance with the previous study [79], we consider love and joy as positive, sadness and fear as negative, and discard the surprise label to avoid introducing noise as surprise can match either positive or negative. For Group-2 samples, the original annotations by three coders were released. We assign each comment with positive, negative, or neutral if it was annotated with corresponding labels by at least two coders. Under such criteria, samples that cannot match any sentiment label are excluded. For each sample in Group-3, its authors directly released its golden emotion labels, which were assigned if at least two raters marked the presence of the emotions [81]. After excluding the surprise labels, we discard the samples with no label or opposite sentiment labels. Finally, we have 2,573 samples remained, 42.9% of which are positive, 27.3% neutral, and 29.8% negative.

Stack Overflow dataset[13] [10] contains 4,423 samples, covering four types of Stack Overflow posts (i.e., questions, answers, question comments, and answer comments). It was originally extracted from the Stack Overflow dump from July 2008 to September 2015. To ensure a balanced polarity distribution, its authors selected 4,800 posts based on their sentiment detected by SentiStrength, and then employed three coders to manually label these posts with positive, negative, or neutral. The posts labeled with opposite polarity labels were excluded by its authors and the final label of the remaining posts were determined via majority voting. Finally, among the remaining 4,423 posts, 34.5% are positive, 38.3% neutral, and the rest 27.2% negative.

[11] http://ansymore.uantwerpen.be/system/files/uploads/artefacts/alessandro/MSR16/archive3.zip.
[12] An issue comment may contain more than one sentence.
[13] https://github.com/collab-uniba/Senti4SD.

Table 2. Benchmark Datasets Used for Sentiment Detection

Dataset	#	Polarity distribution		
		Positive	Neutral	Negative
JIRA	2,573	1,104 (42.9%)	702 (27.3%)	767 (29.8%)
Stack Overflow	4,423	1,527 (34.5%)	1,694 (38.3%)	1,202 (27.2%)
Code Review	1,600	1,202 (75.1%)		398 (24.9%)
Java Library	1,500	131 (8.7%)	1,191 (79.4%)	178 (11.9%)

The *Code Review dataset*[14] [1] contains 1,600 code review comments extracted from code review repositories of 20 popular open source software projects. The dataset originally contained 2,000 comments, each of which was annotated by three coders with positive, negative, or neutral. Based on the annotation results, the distribution of the 2,000 review comments was: 7.7% positive, 19.9% negative, and 72.4% neutral. Due to the serious class imbalance, its authors randomly excluded a subset of the majority class (i.e., neutral) and aggregated the remaining neutral and positive comments into "non-negative" class. Finally, among the remaining 1,600 comments, 24.9% are negative and 75.1% non-negative.

The *Java Library dataset*[15] [61] contains 1,500 sentences about Java libraries/APIs extracted from the Stack Overflow dump of July 2017. Each sentence was annotated by two coders. The coders labeled each sentence with a sentiment score from −2 to 2 (−2 indicates strong negative, -1 weak negative, 0 neutral, 1 weak positive, and 2 strong positive). After the labeling and discussion process, each sentence had a consistent and double-checked sentiment label. Finally, among the 1,500 collected sentences, 8.7% are positive, 79.4% neutral, and 11.9% negative.

We summarize the statistics of the above four datasets in Table 2. Besides them, we also create a unified dataset to further evaluate the generalizability of SEntiMoji and the baseline methods. Since JIRA, Stack Overflow, and Java Library datasets share the same classes (i.e., positive, negative, and neutral), we merge the three datasets into a new dataset, i.e., the *Unified-S dataset*.

4.2.2 Datasets for Emotion Detection. We employ four benchmark datasets (i.e., *JIRA-E1 dataset*, *SO-E dataset*, *JIRA-E2 dataset*, and *Unified-E dataset*) for emotion detection. JIRA-E1, SO-E, and Unified-E datasets are created based on the Shaver framework, while JIRA-E2 is based on the VAD Model. We then describe these datasets briefly:

The *JIRA-E1 dataset*[16] [81] is a cleaned dataset that is extracted from the original 5,992 samples in the JIRA dataset. As mentioned in Section 4.2.1, the 5,992 samples were divided into Group-1, Group-2, and Group-3. Since DEVA includes all the heuristics implemented in SentiStrength-SE that is developed based on the Group-1 samples, to make a fair comparison among different methods, here we do not consider the Group-1 data. In addition, Group-2 data have very imbalanced distribution (e.g., only three samples are labeled with surprise), which is too difficult for a machine learning algorithm to learn a classifier. Therefore, in line with previous studies [11, 12], we exclude the Group-2 data and use only the Group-3 data for emotion detection. Group-3 focuses on four emotions (i.e., love, joy, sadness, and anger). For each emotion, Group-3 has 1,000 sentences labeled with the presence or absence of it. Samples labeled with love, joy, sadness, and anger account for 16.6%, 12.4%, 32.4%, and 30.2% in their corresponding 1,000 samples, respectively.

[14] https://github.com/senticr/SentiCR/.
[15] https://sentiment-se.github.io/replication.zip.
[16] http://ansymore.uantwerpen.be/system/files/uploads/artefacts/alessandro/MSR16/archive3.zip.

Table 3. Benchmark Datasets Used for Emotion Detection: Based on Shaver Framework

Dataset	#	Emotion distribution					
		Love	Joy	Anger	Sadness	Fear	Surprise
JIRA-E1	4*1,000	166 (16.6%)	124 (12.4%)	324 (32.4%)	302 (30.2%)	—	—
SO-E	4,800	1,220 (25.4%)	491 (10.2%)	882 (18.4%)	230 (4.8%)	106 (2.2%)	45 (0.9%)

.

Table 4. A Benchmark Dataset Used for Emotion Detection: Based on VAD Model

Dataset	#	Emotion distribution				
		Excitement	Relaxation	Stress	Depression	Neutral
JIRA-E2	1,795	411 (22.9%)	227 (12.6%)	252 (14.0%)	289 (16.1%)	616 (34.3%)

The *SO-E dataset*[17] [78] is the same dataset as the Stack Overflow dataset, where raters annotated each sample using both emotional labels and sentiment polarity. Specifically, SO-E dataset contains 4,800 posts and focuses on six emotions, including love, joy, anger, sadness, fear, and surprise. Each post was annotated with the presence or absence of each emotion. Finally, 1,959 posts received at least one emotion label. Among the 4,800 samples, 25.4% were labeled with love, 10.2% joy, 18.1% anger, 4.8% sadness, 2.2% fear, and 0.9% surprise.

The *JIRA-E2 dataset*[18] [50] contains 1,795 issue comments extracted from the JIRA issue tracking system. Different from the JIRA-E1 dataset, JIRA-E2 dataset is created based on VAD model [50], where emotions are represented as combinations of different levels of valence and arousal. Specifically, JIRA-E2 dataset involves four emotions, i.e., excitement (positive valence and high arousal), stress (negative valence and high arousal), depression (negative valence and low arousal), and relaxation (positive valence and low arousal). First, its authors used a keyword-based searching method to collect 2,000 samples that were likely to contain valence and arousal from the corpus of the JIRA issue tracking system. Then they employed three coders to annotate each sample with one emotional state based on its valence and arousal level. During the annotation process, 205 issue comments are discarded, since coders cannot achieve an agreement on the emotions perceived in them. Among the remaining 1,795 comments, 22.9% were labeled with excitement, 12.6% relaxation, 14.0% stress, 16.1% depression, and 34.3% neutral.

We summarize the statistics of above three datasets in Table 3 and Table 4. Besides them, we also create a unified dataset for emotion detection. Since JIRA-E1 and SO-E share four classes (i.e., love, joy, anger, and sadness), we merge the two datasets into a new dataset, i.e., the *Unified-E dataset*. Specifically, similar to JIRA-E1, Unified-E contains four subsets. Each subset is for one emotion and contains the corresponding 1,000 samples in JIRA-E1 and all the 4,800 samples in SO-E.

4.3 Evaluation Metrics

In line with previous studies [50, 61], we measure the performance of each method in terms of the *precision*, *recall*, and *F1-score* of each sentiment/emotion class as well as the overall accuracy.

Precision represents the exactness of one method. The precision of a given sentiment/emotion class c is measured as:

$$precision@c = \frac{\#\text{right predicted samples belonging to class } c}{\#\text{total samples predicted as class } c}, \tag{6}$$

[17] https://github.com/collab-uniba/EmotionDatasetMSR18.
[18] https://figshare.com/s/277026f0686f7685b79e.

Recall represents the sensitivity of one method. The recall of a given class c is calculated as:

$$recall@c = \frac{\#\text{right predicted samples belonging to class } c}{\#\text{total samples belonging to class } c}, \tag{7}$$

F1-score is a combination of precision and recall. The F-score of a given class c is computed as the harmonic mean of precision@c and recall@c:

$$F1 - score@c = \frac{2 * precision@c * recall@c}{precision@c + recall@c}. \tag{8}$$

Accuracy measures how often one method makes the correct prediction and is defined as below:

$$accuracy = \frac{\#\text{right predicted samples}}{\#\text{total samples}}, \tag{9}$$

In addition, since some datasets (e.g., Java Library dataset) have a imbalanced class distribution, we also employ three macro-average metrics, which are demonstrated to be suitable for such imbalanced scenarios [91]. Specifically, we employ *macro-precision*, *macro-recall*, and *macro-F1*, which take the average of precision, recall, and F1-score of all classes, respectively. For instance, when we consider the binary classification task that aims to identify a specific emotion from texts, the macro-precision/recall/F1 can be computed as the average of the precision/recall/F1 of the two corresponding classes (i.e., samples containing the emotion and samples not containing the emotion).

In this study, to make a comprehensive comparison, we report the results based on all of the metrics above. However, we urge researchers and practitioners to determine which metrics they should focus on according to their intended goals. For instance, if one needs to perform the sentiment detection in a scenario where precision is important, she may pay more attention to the precision level of different methods. In addition, although all the metrics are well-adopted ones that can help us quickly compare different methods, using them alone may be not enough in some specific cases. For example, in a multi-class emotion detection task, classifying "love" as "joy" may have a different cost than classifying "love" as "sadness," but the adopted metrics cannot reflect such a difference. In this case, we encourage researchers and practitioners to use other methods to complement these metrics. For example, a confusion matrix can be used to show the distribution of the classification results, from which we can observe how many samples with love are misclassified as "joy" and "sadness," respectively.

4.4 Experimental Setting

To make a fair comparison among different methods, for each dataset, we test each method in the same fivefold cross validation setting. More specifically, for a given dataset, we randomly split it into five equal subsets and thus can test each method for five times. Each time, we use one unique subset as the test set and the samples in the remaining four subsets as the training data to train all the methods. For example, each time, we use four subsets to train SEntiMoji, and then test the performance of the obtained SEntiMoji on the remaining subset. Under such an evaluation setting, for each dataset, we calculate the aforementioned metrics of each method for five turns in cross validation and finally report the mean value of each metric.

In addition, several important details need to be clarified:

- Since lexicon-based SentiStrength, SentiStrength-SE, and DEVA are not developed based on supervised machine learning but on a set of rules, we do not re-train them. Instead, for each turn in cross validation, we directly apply the off-the-shelf tools of them to the test set.

- Since DEVA and MarValous aim at identifying excitement, stress, depression, and relaxation, we compare them with other emotion detection methods only on the JIRA-E2 dataset.
- We consider JIRA, Stack Overflow, Java Library, and Unified-S datasets as trinary classification tasks. On each of the four datasets, we train a classifier that can classify texts as positive, negative, or neutral. Since Code Review dataset has only two classes, it is considered as a binary classification task, i.e., negative or non-negative. In addition, as JIRA-E2 dataset focuses on four emotions, we consider it as a multi-class classification task involving five classes, i.e., classifying each sample as excitement, relaxation, stress, depression, or neutral.
- JIRA-E1 and Unified-E datasets both contain four sub-datasets, each of which is for a specific emotion. For example, for the subset for joy, each sample is labeled with joy or not. Therefore, following the previous studies [12, 81], we consider each of the two datasets as four binary classification tasks. In each task, we consider only one emotion and use the corresponding sub-dataset to train a classifier that can judge whether texts contain such an emotion or not.
- SO-E dataset focuses on six emotions and contains samples that are annotated with more than one emotion label. To tackle the complicated multi-label classification, we employ the binary relevance method [87], which transforms the multi-label classification into independently training binary classifiers for each label. The binary relevance method is a common practice for tackling multi-label classification [87] and has been used in previous related work in SE [12]. In line with this work, we consider SO-E dataset as six binary classification tasks. In each task, we consider only one emotion and use the entire SO-E dataset to train a classifier that can judge whether texts contain such an emotion or not.

4.5 Statistical Significance Test

As we employ so many metrics to evaluate each method from different aspects, it is difficult for us to conclude whether one method outperforms the others just based on the difference in one specific metric. To test whether the performance gap between methods is statistically significant, we apply the non-parametric McNemar's test [24]. This test suits well for our purpose as it does not require the normal distribution of data and is also employed in related work [51]. Through the fivefold cross validation, each method has output a predicted label for each sample. To compare method A with method B via McNemar's test, we need to derive the number of samples misclassified by A but not by B (denoted as n_{01}) and the number of samples misclassified by B but not by A (denoted as n_{10}). Then we can compute the statistic $\frac{(|n_{01}-n_{10}|-1)^2}{n_{01}+n_{10}}$, which is distributed as chi-square (χ^2) with one degree of freedom. Since we compare SEntiMoji with existing methods on several benchmark datasets simultaneously (i.e., multi-hypothesis tests), it becomes more likely that we can observe that SEntiMoji outperforms at least one existing method on one dataset [6]. To improve our confidence that such a result can generalize to other data, we need a stricter significance threshold for individual comparisons. To this end, we use Benjamini-Yekutieli procedure [6] for correction. Under such a correction, each individual statistical test becomes stricter, thus making the reported results more reliable. More specifically, we use "*p.adjust*" in R and set "*p.adjust.methods*" as "*BY*" to adjust the p-value of each individual test. The performance difference is considered statistically significant, only if the adjusted p-value of the computed statistic is lower than a pre-specified significance level (usually 0.05).

4.6 Implementation Details

To ensure reproducibility, implementation details of the baseline methods and SEntiMoji are described as follows.

For SentiStrengh, SenStrength-SE, and DEVA, as mentioned before, we directly use the off-the-shelf tools for evaluation. For SentiCR, Senti4SD, and EmoTxt, we train them on the benchmark datasets using the scripts released by their authors. More specifically, for SentiCR, we use the Gradient Boosting Tree (GBT) [31] to reproduce it as recommended by its authors. For EmoTxt, in the training phase, it provides two settings for data sampling, i.e., "*DownSampling*" and "*NoDownSampling*." DownSampling randomly samples training data to make all classes have the same frequency as the minority class, while NoDownSampling does not change the training data. We employ both the two settings for training EmoTxt and denote them as *EmoTxt-Down* and *EmoTxt-No*, respectively. For ESEM-E, we carefully reproduce it following the detailed descriptions in the corresponding paper [74], since its training code has not been released. As recommended by its authors, we use SVM to reproduce ESEM-E. For MarValous, although its training code has also not been released directly, we can find the compiled files of its python scripts in the released package.[19] Therefore, we use a decompiler named *uncompyle*[20] to decompile these files and obtain the training scripts of MarValous. Then we use the obtained scripts to reproduce MarValous and use SVM to train the classifier as recommended by its authors.

For SEntiMoji, SEntiMoji-T, T-80%, T-60%, T-40%, and T-20%, we use the pre-trained DeepMoji as initialization and then fine-tune the model parameters through the chain-thaw approach. During this process, as recommended by Felbo et al. [30], we use Adam optimizer [56] with gradient clipping of the norm to 1 and set learning rate to 10^{-3} for training of the replaced layer and 10^{-4} for fine-tuning any pre-trained layers. For SEntiMoji-G1/2, in stage 1, instead of fine-tuning the pre-trained DeepMoji, we train DeepMoji-SE based on GitHub data from scratch. During this process, SEntiMoji-G1/2 shares the same hyper-parameter settings as DeepMoji. In stage 2, we fine-tune the pre-trained DeepMoji-SE following the same procedure and settings as SEntiMoji. We train and fine-tune all these emoji-based models on a Linux machine with two Tesla M40 GPUs.

4.7 Research Questions

This study aims to answer the following research questions:

RQ1: *How does SEntiMoji perform compared to existing sentiment detection methods in SE?* We aim to verify whether SEntiMoji can achieve better performance than existing sentiment detection methods in SE on the benchmark datasets.

RQ2: *How does SEntiMoji perform compared to existing emotion detection methods in SE?* We aim to verify whether SEntiMoji can achieve better performance than existing emotion detection methods in SE on the benchmark datasets.

RQ3: *Which training corpora contribute more to the power of SEntiMoji?* Since SEntiMoji uses three kinds of corpora (i.e., Twitter data, GitHub data, and manually labeled data), we aim to investigate which corpora contribute more to the power of SEntiMoji.

RQ4: *What are the main difficulties faced by SEntiMoji?* We aim to distill the difficulties faced by SEntiMoji and identify open challenges of sentiment and emotion detection in SE.

5 RESULTS

In this section, we answer our research questions based on the evaluation results.

5.1 RQ1: How Does SEntiMoji Perform Compared to Existing Sentiment Detection Methods in SE?

To answer *RQ1*, we compare SEntiMoji with four existing sentiment detection methods (i.e., SentiStrength, SentiStrength-SE, SentiCR, and Senti4SD) on five benchmark datasets (i.e., JIRA, Stack

[19] https://figshare.com/s/a3308b7087df910db38f.
[20] https://github.com/rocky/python-uncompyle6/.

Overflow, Code Review, Java Library, and Unified-S datasets). We summarize the performance of all the methods in Table 5. For each combination of dataset and metric, we highlight the best result with shading.

Analysis: At a glance of the results in Table 5, we observe that SEntiMoji can achieve the best performance on most metrics. In terms of macro-F1, it can outperform existing methods with an average increase of 0.036 on the five datasets. Next, we analyze the results thoroughly.

We first compare SEntiMoji with the most widely used SentiStrength, which is an off-the-shelf sentiment detection tool without any SE-customized efforts. On JIRA, Stack Overflow, Code Review, Java Library, and Unified-S datasets, the macro-F1 obtained by SEntiMoji is 0.154, 0.060, 0.168, 0.185, and 0.078 higher than that obtained by SentiStrength, respectively. By comparison, the difference on the Stack Overflow dataset is the smallest, only 0.060. This "outlier" can be attributed to the creation process of the Stack Overflow dataset. Calefato et al. [10] created this dataset by sampling the originally collected posts based on their sentiment scores computed by SentiStrength. It is easier for SentiStrength to correctly classify the samples selected by itself, which thus results in a relatively small performance gap between SEntiMoji and SentiStrength on the Stack Overflow dataset.

Then we want to compare SEntiMoji with the SE-customized methods (i.e., SentiStrength-SE, SentiCR, and Senti4SD). In general, SentiCR performs the best among the three existing methods as it can achieve the highest accuracy and macro-F1 on each dataset except the Stack Overflow and Unified-S datasets. On the Stack Overflow and Unified-S datasets, it performs slightly worse than Senti4SD. Similarly, Islam and Zibran [49] also found that Senti4SD can achieve the highest accuracy on Stack Overflow dataset. It is reasonable that the semantic features used by Senti4SD are extracted based on the embeddings trained on a large-scale Stack Overflow corpus, and thus Senti4SD is more knowledgeable than SentiCR when dealing with Stack Overflow posts in Stack Overflow and Unified-S datasets. As SentiCR has an obvious advantage over other existing SE-customized methods in general, we then just compare SEntiMoji with it.

Compared to SentiCR, SEntiMoji obtains a better result in 54 of 62 metrics. In terms of macro-F1, SEntiMoji outperforms SentiCR on all the datasets, with an average increase of 0.044. Similarly, SEntiMoji can achieve a higher accuracy level. For example, on the JIRA dataset, the accuracy scores obtained by SEntiMoji and SentiCR are 0.904 and 0.872, respectively. In other words, their error rates are 0.096 and 0.128, respectively. It indicates that SEntiMoji can reduce 25% of the samples misclassified by SentiCR on the JIRA dataset. In addition, when carefully inspecting the results, we find that the performance gap between SentiCR and SEntiMoji is particularly large in some cases. For instance, in terms of precision, the extent to which SEntiMoji outperforms SentiCR on the Java Library dataset is obviously larger than on other datasets. This phenomenon can be attributed to the sampling methods of different datasets. As JIRA dataset is labeled with various emotion labels, we need to map the multi-class emotions into trinary polarities and some samples are filtered due to ambiguity, which results in a relatively balanced data distribution. With regard to Stack Overflow and Code Review datasets, we find both of them were manually pre-processed and filtered to follow a not that skewed sentiment distribution during their creation process [1, 10]. The sampling of these datasets makes the classification tasks considerably easier, so SentiCR and SEntiMoji do not show an obvious performance difference on these datasets. Compared to other datasets, Java Library dataset has a more imbalanced class distribution, i.e., 79.4% of samples are neutral. In such a situation, SEntiMoji can achieve 0.296 and 0.183 higher precision over positive and negative samples while keeping the similar level of recall, which further demonstrates the superiority of SEntiMoji.

Finally, to verify whether the superiority of SEntiMoji is statistically significant, for each dataset, we perform McNemar's test between the results produced by SEntiMoji and each baseline method.

Table 5. Performance of SEntiMoji and Existing Sentiment Detection Methods

Dataset	Class	Metric	SentiStrength	SentiStrength-SE	SentiCR	Senti4SD	SEntiMoji
JIRA	Pos	Precision	0.847	0.936	**0.950**	0.880	0.947
		Recall	0.889	0.922	0.919	0.921	**0.945**
		F1-score	0.868	0.929	0.934	0.900	**0.946**
	Neu	Precision	0.614	0.710	0.735	0.741	**0.823**
		Recall	0.634	0.844	**0.904**	0.731	0.880
		F1-score	0.623	0.771	0.811	0.736	**0.850**
	Neg	Precision	0.775	0.871	**0.929**	0.835	0.922
		Recall	0.699	0.734	0.768	0.789	**0.864**
		F1-score	0.735	0.796	0.840	0.811	**0.892**
	Accuracy		0.763	0.846	0.872	0.830	**0.904**
	Macro-precision		0.746	0.839	0.871	0.819	**0.897**
	Macro-recall		0.740	0.833	0.864	0.814	**0.896**
	Macro-F1		0.742	0.832	0.862	0.816	**0.896**
Stack Overflow	Pos	Precision	0.887	0.908	0.868	0.904	**0.932**
		Recall	0.927	0.823	0.921	0.915	**0.940**
		F1-score	0.907	0.863	0.894	0.910	**0.936**
	Neu	Precision	**0.922**	0.726	0.783	0.829	0.840
		Recall	0.632	0.784	0.838	0.772	**0.842**
		F1-score	0.750	0.754	0.809	0.800	**0.841**
	Neg	Precision	0.674	0.755	0.843	0.778	**0.846**
		Recall	**0.931**	0.759	0.686	0.841	0.833
		F1-score	0.780	0.757	0.753	0.808	**0.838**
	Accuracy		0.815	0.800	0.826	0.840	**0.873**
	Macro-precision		0.827	0.804	0.829	0.837	**0.873**
	Macro-recall		0.830	0.797	0.815	0.843	**0.872**
	Macro-F1		0.812	0.798	0.819	0.839	**0.872**
Code Review	Non-Neg	Precision	0.806	0.795	**0.872**	0.840	0.869
		Recall	0.814	0.919	0.895	0.912	**0.941**
		F1-score	0.809	0.852	0.883	0.875	**0.904**
	Neg	Precision	0.506	0.537	0.660	0.638	**0.762**
		Recall	0.474	0.238	**0.600**	0.475	0.572
		F1-score	0.488	0.372	0.627	0.544	**0.653**
	Accuracy		0.712	0.761	0.823	0.804	**0.849**
	Macro-precision		0.612	0.666	0.766	0.739	**0.816**
	Macro-recall		0.610	0.602	0.748	0.693	**0.756**
	Macro-F1		0.610	0.612	0.755	0.709	**0.778**
Java Library	Pos	Precision	0.202	0.320	0.553	0.472	**0.849**
		Recall	**0.369**	0.224	0.318	0.203	0.329
		F1-score	0.206	0.262	0.401	0.266	**0.472**
	Neu	Precision	0.858	0.824	**0.883**	0.860	0.880
		Recall	0.768	0.929	0.910	0.926	**0.964**
		F1-score	0.810	0.873	0.896	0.893	**0.920**
	Neg	Precision	0.396	0.487	0.546	0.522	**0.729**
		Recall	0.434	0.183	**0.593**	0.463	0.583
		F1-score	0.412	0.265	0.565	0.481	**0.644**
	Accuracy		0.693	0.778	0.821	0.807	**0.863**
	Macro-precision		0.485	0.544	0.661	0.618	**0.819**
	Macro-recall		0.524	0.445	0.607	0.531	**0.625**
	Macro-F1		0.494	0.467	0.621	0.546	**0.679**

(Continued)

Table 5. Continued

Dataset	Class	Metric	SentiStrength	SentiStrength-SE	SentiCR	Senti4SD	SEntiMoji
Unified-S	Pos	Precision	0.805	0.899	0.904	0.875	**0.920**
		Recall	0.866	0.834	0.862	0.878	**0.897**
		F1-score	0.834	0.865	0.882	0.877	**0.909**
	Neu	Precision	0.798	0.755	0.771	0.805	**0.824**
		Recall	0.673	0.844	**0.873**	0.809	0.867
		F1-score	0.730	0.797	0.819	0.807	**0.845**
	Neg	Precision	0.670	0.786	0.795	0.771	**0.838**
		Recall	0.780	0.704	0.666	0.761	**0.790**
		F1-score	0.721	0.742	0.723	0.765	**0.812**
	Accuracy		0.763	0.805	0.817	0.819	**0.857**
	Macro-precision		0.762	0.801	0.808	0.816	**0.855**
	Macro-recall		0.758	0.813	0.823	0.817	**0.861**
	Macro-F1		0.773	0.794	0.800	0.816	**0.851**

Note: For each metric, the highest value is highlighted with shading.

Table 6. McNemar's Statistics between the Results of SEntiMoji and Other Existing Sentiment Detection Methods on Different Datasets, with Adjusted p-values in Parentheses

	SentiStrength	SentiStrength-SE	SentiCR	Senti4SD
JIRA	231.843 (0.000)	62.510 (0.000)	23.616 (0.000)	85.708 (0.000)
Stack Overflow	95.394 (0.000)	128.548 (0.000)	76.963 (0.000)	38.686 (0.000)
Code Review	102.202 (0.000)	55.840 (0.000)	7.320 (0.025)	19.711 (0.000)
Java Library	160.160 (0.000)	69.522 (0.000)	20.556 (0.000)	40.786 (0.000)
Unified-S	377.202 (0.000)	154.756 (0.000)	99.070 (0.000)	91.187 (0.000)

Note: All the reported results in this table are significant at 5% level.

We present the statistic results in Table 6. For each dataset, we formulate null and alternative hypotheses to determine the statistical significance of the performance gap between SEntiMoji and existing sentiment detection methods on it. Take the JIRA dataset as an example. The corresponding null and alternative hypotheses are as follows:

Null hypothesis-1 (H_0^1): There is no significant difference in the performance of SEntiMoji compared to existing sentiment detection methods on the JIRA dataset.

Alternative hypothesis-1 (H_a^1): There exist significant differences in the performance of SEntiMoji compared to existing sentiment detection methods on the JIRA dataset.

According to Table 6, on the JIRA dataset, the performance difference between SEntiMoji and each baseline method is found to be statistically significant with $p = 0.000$ and $p < 0.05$. Therefore, the McNemar's test rejects our null hypothesis (H_0^1), and the alternative hypothesis (H_a^1) holds true. Similarly, McNemar's test can reject the null hypotheses formulated for the remaining four benchmark datasets at a significance level of 0.05. Therefore, we can derive the answer to *RQ1* as follows:

Ans. to RQ1: SEntiMoji can significantly outperform existing sentiment detection methods in SE on all of the benchmark datasets.

5.2 RQ2: How Does SEntiMoji Perform Compared to Existing Emotion Detection Methods in SE?

To answer *RQ2*, we compare SEntiMoji with five existing emotion detection methods (i.e., DEVA, EmoTxt-Down, EmoTxt-No, MarValous, and ESEM-E) on JIRA-E1, SO-E, JIRA-E2, and

Unified-E datasets. As explained in Section 4.4, we evaluate DEVA and MarValous only on the JIRA-E2 dataset, and consider the JIRA-E1, SO-E, and Unified-E datasets as four, six, and four binary classification tasks, respectively. We summarize the performance of all these methods in Table 7. For each combination of dataset and metric, we highlight the best result with shading. To save space, for each task on the JIRA-E1, SO-E, and Unified-E datasets, we report only the precision, recall, F1-score of the detected emotions as well as the overall accuracy, macro-precision, macro-recall, and macro-F1.

Analysis: At a glance of the results in Table 7, we can find that SEntiMoji achieves the highest value on most metrics. In particular, SEntiMoji obtains the highest accuracy in all the tasks and the highest macro-F1 in 14 of 15 tasks. In terms of macro-F1, it can outperform these methods with an average increase of 0.036. Next, we analyze the results thoroughly.

First, we compare SEntiMoji with DEVA and MarValous. SEntiMoji can outperform DEVA in terms of all the 19 reported metrics on the JIRA-E2 dataset. Given the accuracy scores of DEVA and SEntiMoji are 0.830 and 0.886, their error rates are 0.170 and 0.114, respectively. In other words, SEntiMoji can reduce 32.9% of the errors compared to DEVA. Similarly, SEntiMoji can outperform MarValous in 18 of 19 metrics on the JIRA-E2 dataset.

Then we compare SEntiMoji with EmoTxt-Down. An obvious advantage of EmoTxt-Down is that it achieves the highest recall of emotions in 7 of 10 tasks of JIRA-E1 and SO-E. It is reasonable, since EmoTxt-Down samples training data to ensure a balanced distribution of different classes. By comparison, SEntiMoji is trained based on imbalanced data and thus tends to classify samples as the majority class (i.e., absence of the emotion). As a result, SEntiMoji cannot find as many emotional samples as EmoTxt-Down in some cases. However, the disadvantage of EmoTxt-Down is also very obvious: its precision over emotional samples is often much lower than SEntiMoji. For example, in the sadness detection task of the SO-E dataset, SEntiMoji obtains more than three times the precision of sadness samples obtained by EmoTxt-Down. The disadvantage of EmoTxt-Down can be attributed to its sampling strategy, too. As emotional samples are scarce in the JIRA-E1 and SO-E datasets, to achieve a balanced distribution, EmoTxt-Down needs to filter out a large amount of nonemotional samples. Therefore, it lacks knowledge about nonemotional samples and has a high chance of classifying nonemotional samples as emotional, which results in a low precision on emotional samples. By comparison, SEntiMoji can achieve a better balance between precision and recall (i.e., higher F1-score) than EmoTxt-Down.

Next, we compare SEntiMoji with EmoTxt-No. We find that SEntiMoji can outperform EmoTxt-No in almost all the metrics. The main disadvantage of EmoTxt-No is its low recall level, especially on the JIRA-E1 and SO-E datasets. For example, in the love detection task of the JIRA-E1 dataset, the recall level of love samples obtained by EmoTxt-No is only 0.066, while the one obtained by SEntiMoji is 0.674. In the surprise detection task of the SO-E dataset, EmoTxt-No even has no ability to distinguish between surprise and no-surprise as it classifies all the test samples as no-surprise.[21] The low recall of EmoTxt-No can be attributed to the scarcity of emotional samples in these two datasets, which causes that EmoTxt-No lacks knowledge about emotional samples and thus can identify only a few emotional samples. By comparison, since SEntiMoji is developed based on large-scale emoji samples that contain rich emotional information, it can alleviate this shortcoming to some extent.

Finally, we compare SEntiMoji with ESEM-E. Although ESEM-E achieves a higher macro-F1 score than SEntiMoji in the surprise task of SO-E, it achieves lower macro-F1 scores than SEntiMoji in all the other tasks. In addition, SEntiMoji achieves higher accuracy than ESEM-E in all the tasks.

[21] Since all the test samples are classified as no-surprise, we cannot calculate the precision and F1-score of surprise class.

Table 7. Performance of SEntiMoji and Existing Emotion Detection Methods

Dataset	Class	Metric	DEVA	EmoTxt-Down	EmoTxt-No	MarValous	ESEM-E	SEntiMoji
JIRA-E1	Love	Precision	-	0.377	0.316	-	0.663	**0.713**
		Recall	-	**0.936**	0.066	-	0.390	0.674
		F1-score	-	0.531	0.103	-	0.483	**0.692**
		Accuracy	-	0.740	0.830	-	0.865	**0.907**
		Macro-precision	-	0.679	0.578	-	0.774	**0.825**
		Macro-recall	-	**0.819**	0.526	-	0.676	0.813
		Macro-F1	-	0.674	0.504	-	0.703	**0.818**
	Joy	Precision	-	0.397	**0.820**	-	0.728	0.780
		Recall	-	**0.753**	0.291	-	0.421	0.503
		F1-score	-	0.489	0.426	-	0.528	**0.606**
		Accuracy	-	0.790	0.904	-	0.911	**0.921**
		Macro-precision	-	0.680	**0.864**	-	0.826	0.857
		Macro-recall	-	**0.774**	0.641	-	0.700	0.741
		Macro-F1	-	0.676	0.687	-	0.739	**0.781**
	Anger	Precision	-	0.622	0.761	-	0.739	**0.835**
		Recall	-	0.642	0.503	-	0.554	**0.664**
		F1-score	-	0.603	0.566	-	0.633	**0.739**
		Accuracy	-	0.718	0.755	-	0.784	**0.843**
		Macro-precision	-	0.720	0.776	-	0.769	**0.840**
		Macro-recall	-	0.694	0.689	-	0.727	**0.798**
		Macro-F1	-	0.683	0.691	-	0.740	**0.813**
	Sadness	Precision	-	0.822	**0.976**	-	0.859	0.931
		Recall	-	0.714	0.636	-	0.607	**0.765**
		F1-score	-	0.741	0.769	-	0.709	**0.838**
		Accuracy	-	0.843	0.885	-	0.851	**0.911**
		Macro-precision	-	0.852	0.920	-	0.854	**0.918**
		Macro-recall	-	0.807	0.815	-	0.782	**0.869**
		Macro-F1	-	0.813	0.846	-	0.805	**0.888**
SO-E	Love	Precision	-	0.669	0.803	-	0.786	**0.807**
		Recall	-	0.781	0.584	-	0.645	**0.814**
		F1-score	-	0.702	0.676	-	0.708	**0.810**
		Accuracy	-	0.846	0.858	-	0.865	**0.903**
		Macro-precision	-	0.795	0.836	-	0.836	**0.872**
		Macro-recall	-	0.825	0.767	-	0.793	**0.874**
		Macro-F1	-	0.807	0.792	-	0.810	**0.872**
	Joy	Precision	-	0.271	0.712	-	0.471	**0.744**
		Recall	-	**0.607**	0.184	-	0.338	0.379
		F1-score	-	0.374	0.291	-	0.392	**0.475**
		Accuracy	-	0.792	0.908	-	0.893	**0.913**
		Macro-precision	-	0.795	0.836	-	0.699	**0.872**
		Macro-recall	-	0.825	0.767	-	0.647	**0.874**
		Macro-F1	-	0.807	0.792	-	0.667	**0.872**
	Anger	Precision	-	0.486	0.778	-	0.682	**0.812**
		Recall	-	**0.690**	0.480	-	0.456	0.677
		F1-score	-	0.570	0.593	-	0.547	**0.738**
		Accuracy	-	0.809	0.880	-	0.861	**0.912**
		Macro-precision	-	0.705	0.835	-	0.784	**0.871**
		Macro-recall	-	0.763	0.725	-	0.704	**0.821**
		Macro-F1	-	0.724	0.761	-	0.732	**0.842**
	Sadness	Precision	-	0.230	0.783	-	0.580	**0.794**
		Recall	-	**0.700**	0.225	-	0.364	0.455
		F1-score	-	0.344	0.347	-	0.447	**0.577**
		Accuracy	-	0.871	0.960	-	0.957	**0.968**
		Macro-precision	-	0.607	0.872	-	0.774	**0.884**
		Macro-recall	-	**0.790**	0.611	-	0.675	0.725
		Macro-F1	-	0.636	0.663	-	0.712	**0.780**
	Fear	Precision	-	0.129	**0.800**	-	0.617	0.682
		Recall	-	**0.808**	0.046	-	0.162	0.296
		F1-score	-	0.220	0.087	-	0.245	**0.404**
		Accuracy	-	0.878	0.979	-	0.979	**0.981**
		Macro-precision	-	0.562	**0.889**	-	0.799	0.833
		Macro-recall	-	**0.844**	0.523	-	0.580	0.646
		Macro-F1	-	0.577	0.538	-	0.617	**0.697**
	Surprise	Precision	-	0.018	-	-	**0.467**	0.400
		Recall	-	**0.615**	0.000	-	0.063	0.045
		F1-score	-	0.034	-	-	**0.109**	0.081
		Accuracy	-	0.570	0.991	-	0.990	**0.993**

(Continued.)

Table 7. Continued

		Macro-precision	-	0.506	-	-	**0.729**	0.696
		Macro-recall	-	**0.592**	0.500	-	0.531	0.522
		Macro-F1	-	0.362	-	-	**0.552**	0.538
JIRA-E2	Excitement	Precision	0.876	0.844	0.858	0.889	0.874	**0.904**
		Recall	0.894	0.826	0.841	0.946	**0.949**	0.942
		F1-score	0.884	0.834	0.849	0.917	0.909	**0.922**
	Relaxation	Precision	0.855	0.648	0.824	0.828	0.834	**0.858**
		Recall	0.656	0.768	0.585	0.741	0.800	**0.814**
		F1-score	0.742	0.693	0.682	0.778	0.816	**0.834**
	Stress	Precision	0.723	0.677	0.839	0.793	**0.881**	0.853
		Recall	0.658	0.579	0.473	0.482	0.573	**0.721**
		F1-score	0.688	0.622	0.604	0.595	0.693	**0.780**
	Depression	Precision	0.780	0.713	0.823	0.812	**0.902**	0.846
		Recall	0.754	0.793	0.637	0.781	0.792	**0.841**
		F1-score	0.766	0.744	0.716	0.795	**0.843**	0.842
	Neutral	Precision	0.852	0.846	0.683	0.815	0.832	**0.911**
		Recall	0.956	0.768	0.954	0.950	0.964	**0.965**
		F1-score	0.901	0.794	0.795	0.877	0.893	**0.937**
	Accuracy		0.830	0.759	0.764	0.830	0.857	**0.886**
	Macro-precision		0.817	0.746	0.806	0.828	0.865	**0.875**
	Macro-recall		0.783	0.747	0.698	0.780	0.816	**0.856**
	Macro-F1		0.796	0.737	0.729	0.792	0.831	**0.863**
Unified-E	Love	Precision	-	0.640	0.798	-	0.772	**0.828**
		Recall	-	**0.763**	0.504	-	0.700	0.726
		F1-score	-	0.696	0.618	-	0.734	**0.774**
		Accuracy	-	0.842	0.853	-	0.879	**0.907**
		Macro-precision	-	0.784	0.821	-	0.840	**0.871**
		Macro-recall	-	0.827	0.747	-	0.817	**0.875**
		Macro-F1	-	0.800	0.773	-	0.828	**0.872**
	Joy	Precision	-	0.339	**0.848**	-	0.549	0.739
		Recall	-	**0.649**	0.214	-	0.356	0.372
		F1-score	-	0.445	0.341	-	0.431	**0.495**
		Accuracy	-	0.793	0.906	-	0.901	**0.913**
		Macro-precision	-	0.619	**0.804**	-	0.738	0.802
		Macro-recall	-	**0.727**	0.592	-	0.660	0.663
		Macro-F1	-	0.635	0.625	-	0.688	**0.705**
	Anger	Precision	-	0.573	0.804	-	0.761	**0.831**
		Recall	-	**0.717**	0.514	-	0.614	0.652
		F1-score	-	0.637	0.627	-	0.679	**0.731**
		Accuracy	-	0.812	0.864	-	0.878	**0.896**
		Macro-precision	-	0.727	0.830	-	0.831	**0.859**
		Macro-recall	-	0.769	0.726	-	0.781	**0.814**
		Macro-F1	-	0.742	0.759	-	0.802	**0.833**
	Sadness	Precision	-	0.589	**0.889**	-	0.805	0.886
		Recall	-	0.673	0.566	-	0.561	**0.678**
		F1-score	-	0.628	0.692	-	0.661	**0.768**
		Accuracy	-	0.926	0.953	-	0.947	**0.955**
		Macro-precision	-	0.781	**0.926**	-	0.881	0.913
		Macro-recall	-	**0.807**	0.772	-	0.774	0.798
		Macro-F1	-	0.789	0.825	-	0.816	**0.843**

Note: For each metric, the highest value is highlighted with shading.

To verify whether the superiority of SEntiMoji is statistically significant, for each task, we perform McNemar's test between the results produced by SEntiMoji and each baseline method. We present the statistic results in Table 8. Then, for each task, we formulate null and alternative hypotheses to determine the statistical significance of the performance gap between SEntiMoji and existing emotion detection methods. Take the love detection task of the JIRA-E1 dataset as an example. The null and alternative hypotheses formulated for it are as follows:

Null hypothesis-2 (H_0^2): There is no significant difference in the performance of SEntiMoji compared to existing emotion detection methods in the love detection task of the JIRA-E1 dataset.

Alternative hypothesis-2 (H_a^2): There exist significant differences in the performance of SEntiMoji compared to existing emotion detection methods in the love detection task of the JIRA-E1 dataset.

Table 8. McNemar's Statistics between the Results of SEntiMoji and Other Existing Emotion Methods on Different Datasets, with Adjusted p-values in Parentheses

	DEVA	EmoTxt-Down	EmoTxt-No	MarValous	ESEM-E
JIRA-E1 (Love)	-	**110.667 (0.000)**	**41.554 (0.000)**	-	**13.779 (0.000)**
JIRA-E1 (Joy)	-	**77.880 (0.000)**	4.655 (0.154)	-	1.446 (1.000)
JIRA-E1 (Anger)	-	**64.996 (0.000)**	**38.844 (0.000)**	-	**19.572 (0.000)**
JIRA-E1 (Sadness)	-	**38.042 (0.000)**	**9.766 (0.009)**	-	**29.500 (0.000)**
SO-E (Love)	-	**117.926 (0.000)**	**79.812 (0.000)**	-	**59.135 (0.000)**
SO-E (Joy)	-	**358.605 (0.000)**	**7.006 (0.011)**	-	**37.748 (0.000)**
SO-E (Anger)	-	**296.168 (0.000)**	**54.951 (0.000)**	-	**113.717 (0.000)**
SO-E (Sadness)	-	**366.483 (0.000)**	**17.686 (0.000)**	-	**23.805 (0.000)**
SO-E (Fear)	-	**406.638 (0.000)**	2.439 (0.572)	-	1.961 (0.763)
SO-E (Surprise)	-	**1964.459 (0.000)**	0.000 (1.000)	-	0.167 (1.000)
JIRA-E2	**33.670 (0.000)**	**139.268 (0.000)**	**137.751 (0.000)**	**39.204 (0.000)**	**12.505 (0.000)**
Unified-E (Love)	-	**187.258 (0.000)**	**137.597 (0.000)**	-	**46.942 (0.000)**
Unified-E (Joy)	-	**423.398 (0.000)**	**13.900 (0.000)**	-	**16.150 (0.000)**
Unified-E (Anger)	-	**243.892 (0.000)**	**55.684 (0.000)**	-	**19.924 (0.000)**
Unified-E (Sadness)	-	**95.675 (0.000)**	**7.562 (0.031)**	-	**13.170 (0.000)**

Note: Results significant at 5% level are highlighted with shading.

According to Table 8, in the love detection task of the JIRA-E1 dataset, the performance difference between SEntiMoji and each baseline method is found to be statistically significant with $p = 0.000$ and $p < 0.05$. Thus, the McNemar's test rejects the null hypothesis (H_0^2), and the alternative hypothesis (H_a^2) holds true. It indicates that SEntiMoji can significantly outperform the existing methods in the love detection task of the JIRA-E1 dataset. Similarly, McNemar's test can reject the null hypotheses formulated for all the remaining tasks except the joy detection task of the JIRA-E1 dataset and the fear and surprise detection tasks of the SO-E dataset. In the JIRA-E1 dataset, samples labeled with joy account for only 12.4%, less than the ones labeled with other three emotions. Such a distribution makes the joy detection task difficult and thus SEntiMoji cannot show a significant difference from existing methods. Similarly, the fear and surprise detection tasks are the most difficult ones on the SO-E dataset, as samples labeled with the two emotions account for only 2.2% and 0.9%, respectively. Take the surprise detection task as an example. Given that more than 99% of samples are labeled with no-surprise, the classification task is like "finding needles in a haystack." In such a situation, EmoTxt-No even has no ability to distinguish between surprise and no-surprise. Although SEntiMoji does not perform as badly as EmoTxt-No, SEntiMoji cannot achieve significantly better results than other methods due to the difficulty of the problem. Based on the aforementioned analysis, we can derive an answer to RQ2:

Ans. to RQ2: SEntiMoji can significantly outperform existing emotion detection methods in SE, unless the data distribution is very imbalanced.

5.3 RQ3: Which Training Corpora Contribute More to the Power of SEntiMoji?

Since SEntiMoji can achieve significant improvement on most benchmark datasets, we then investigate the reasons behind its power, which can provide insights for future research. As SEntiMoji does not outperform other methods significantly in the joy detection task of the JIRA-E1 dataset and the fear and surprise detection tasks of the SO-E dataset, in this part, we do not take the three tasks into consideration. In the remaining tasks, we evaluate SEntiMoji and its variants to measure the contribution of different training corpora.

5.3.1 GitHub Data vs. Twitter Data. We first compare SEntiMoji with SEntiMoji-G1, SEntiMoji-G2, and SEntiMoji-T, to measure the contribution of GitHub data and Twitter data. The four

methods share the same model architecture and use the same manually labeled data. They differ only in the external data used in representation learning. Specifically, SEntiMoji uses Twitter data to supply the general sentimental expressions and uses GitHub data to learn technical knowledge. By comparison, SEntiMoji-G1 and SEntiMoji-G2 uses only GitHub data, while SEntiMoji-T uses only Twitter data. We summarize the performance of these methods in Table 9 and Table 10. In addition, to evaluate the statistical significance of the performance difference, we perform the McNemar's test to compare SEntiMoji-G1, SEntiMoji-G2, and SEntiMoji-T with SEntiMoji. Table 11 presents the statistic results. Next, we analyze the results thoroughly.

Analysis for sentiment detection: Table 9 presents the performance of SEntiMoji and its variants in sentiment detection tasks. We can observe that SEntiMoji-T obtains comparable results with SEntiMoji, with only an average decrease of 0.013 in macro-F1 on the five datasets. On the Unified-S dataset, SEntiMoji-T can even obtain a slightly higher macro-F1 score than SEntiMoji. The comparable performance of SEntiMoji and SEntiMoji-T is also demonstrated by the results of statistical tests. As shown in Table 11, the the performance difference between SEntiMoji and SEntiMoji-T is not statistically significant on any of the five datasets for sentiment detection.

By comparison, the performance degradation of SEntiMoji-G1 and SEntiMoji-G2 compared to SEntiMoji is more obvious. Specifically, the average decrease of SEntiMoji-G1 and SEntiMoji-G2 in macro-F1 is 0.091 and 0.174, 7 times and 13 times that of SEntiMoji-T, respectively. In addition, SEntiMoji-G1 and SEntiMoji-G2 lose the advantage of our emoji-powered method in the precision level on the Java Library dataset. The precision@neg gained by SEntiMoji-G1 and SEntiMoji-G2 is 0.489 and 0.311, much lower than the precision levels obtained by SEntiMoji (0.729) and SEntiMoji-T (0.734). Moreover, statistic results in Table 11 show that SEntiMoji can significantly outperform SEntiMoji-G1 and SEntiMoji-G2 on all the five datasets for sentiment detection with $p = 0.000$.

Analysis for emotion detection: Table 10 presents the performance of SEntiMoji and its variants in emotion detection tasks. In these tasks, SEntiMoji-T achieves comparable results with SEntiMoji. In terms of macro-F1, SEntiMoji-T achieves better performance than SEntiMoji in three tasks, with an average increase of 0.008. In the remaining nine tasks, SEntiMoji-T achieves slight worse performance, with an average decrease of only 0.009 in macro-F1 compared to SEntiMoji. The results of statistical tests in Table 11 further demonstrate that the performance difference between SEntiMoji and SEntiMoji-T is not significant in 11 of 12 emotion detection tasks.

By comparison, SEntiMoji-G1 and SEntiMoji-G2 performs obviously worse than SEntiMoji. For example, on the JIRA-E2 dataset, SEntiMoji-G1, and SEntiMoji-G2 achieve the macro-F1 scores of 0.596 and 0.569, which are 0.267 and 0.294 lower than the macro-F1 achieved by SEntiMoji (0.863), respectively. In addition, results in Table 11 demonstrate that in 11 of 12 tasks, the performance gap between SEntiMoji and SEntiMoji-G1/G2 is statistically significant.

Based on the above analysis for sentiment and emotion detection, we can find that SEntiMoji can obtain comparable performance with SEntiMoji-T. However, the performance SEntiMoji-G1/2 is significantly worse than SEntiMoji. Therefore, we can conclude that Twitter data contribute more than GitHub data to the power of SEntiMoji.

5.3.2 Contribution of Manually Labeled Data. We then measure the contribution of the manually labeled data. Specifically, we compare SEntiMoji-T with its variants (i.e., T-80%, T-60%, T-40%, and T-20%) to verify whether the performance declines with the reduction of the manually labeled data. All the five methods use only Tweets for representation learning, and learn technical knowledge only from the labeled data. The sole difference is the amount of the labeled data that they used. The performance of these methods are summarized in Table 12 and Table 13. In addition, we perform the McNemar's test to evaluate the statistical significance of the performance difference between SEntiMoji-T and its variants. The results of statistical tests are presented in Table 14. Next, we analyze the results in detail.

Table 9. Performance of SEntiMoji and Its Variants in Sentiment Detection Tasks

Dataset	Class	Metric	SEntiMoji	SEntiMoji-G1	SEntiMoji-G2	SEntiMoji-T
JIRA	Pos	Precision	**0.947**	0.938	0.885	0.939
		Recall	**0.945**	0.920	0.868	0.933
		F1-score	**0.946**	0.928	0.876	0.936
	Neu	Precision	**0.823**	0.780	0.707	0.808
		Recall	**0.880**	0.869	0.752	0.866
		F1-score	**0.850**	0.822	0.729	0.845
	Neg	Precision	**0.922**	0.888	0.808	0.913
		Recall	**0.864**	0.818	0.782	0.839
		F1-score	**0.892**	0.851	0.795	0.874
	Accuracy		**0.904**	0.876	0.811	0.893
	Macro-precision		**0.897**	0.869	0.800	0.887
	Macro-recall		**0.896**	0.869	0.801	0.886
	Macro-F1		**0.896**	0.867	0.800	0.885
Stack Overflow	Pos	Precision	**0.932**	0.908	0.794	0.921
		Recall	**0.940**	0.864	0.782	0.937
		F1-score	**0.936**	0.885	0.788	0.929
	Neu	Precision	0.840	0.753	0.667	**0.841**
		Recall	**0.842**	0.805	0.733	0.840
		F1-score	**0.841**	0.778	0.698	**0.841**
	Neg	Precision	0.846	0.762	0.694	**0.854**
		Recall	0.833	0.736	0.612	**0.838**
		F1-score	0.838	0.748	0.650	**0.845**
	Accuracy		**0.873**	0.806	0.717	**0.873**
	Macro-precision		**0.873**	0.808	0.718	0.872
	Macro-recall		**0.872**	0.801	0.709	**0.872**
	Macro-F1		**0.872**	0.804	0.712	**0.872**
Code Review	Non-Neg	Precision	**0.869**	0.801	0.778	0.850
		Recall	0.941	0.943	0.939	**0.958**
		F1-score	**0.904**	0.865	0.851	0.900
	Neg	Precision	0.762	0.633	0.507	**0.793**
		Recall	**0.572**	0.291	0.188	0.491
		F1-score	**0.653**	0.393	0.275	0.603
	Accuracy		**0.849**	0.780	0.752	0.841
	Macro-precision		0.816	0.717	0.642	**0.821**
	Macro-recall		**0.756**	0.617	0.564	0.724
	Macro-F1		**0.778**	0.629	0.563	0.752
Java Library	Pos	Precision	0.849	0.737	0.643	**0.878**
		Recall	**0.329**	0.230	0.069	0.281
		F1-score	**0.472**	0.349	0.124	0.426
	Neu	Precision	**0.880**	0.834	0.808	0.869
		Recall	0.964	0.967	0.977	0.971
		F1-score	**0.920**	0.895	0.884	0.917
	Neg	Precision	0.729	0.489	0.311	**0.734**
		Recall	**0.583**	0.223	0.079	0.513
		F1-score	**0.644**	0.302	0.126	0.599
	Accuracy		**0.863**	0.814	0.791	0.859
	Macro-precision		0.819	0.687	0.587	**0.827**
	Macro-recall		**0.625**	0.473	0.375	0.589
	Macro-F1		**0.679**	0.516	0.378	0.647
Unified-S	Pos	Precision	**0.920**	0.875	0.861	0.914
		Recall	**0.897**	0.840	0.811	**0.897**
		F1-score	**0.909**	0.857	0.835	0.906
	Neu	Precision	0.824	0.765	0.719	**0.829**
		Recall	**0.867**	0.817	0.833	0.865
		F1-score	0.845	0.789	0.771	**0.846**
	Neg	Precision	0.838	0.803	0.744	**0.843**
		Recall	0.790	0.752	0.602	**0.801**
		F1-score	0.812	0.776	0.665	**0.820**
	Accuracy		0.857	0.808	0.767	**0.859**
	Macro-precision		0.861	0.814	0.774	**0.862**
	Macro-recall		0.851	0.803	0.749	**0.854**
	Macro-F1		0.855	0.808	0.757	**0.857**

Note: For each metric, the highest value is highlighted with shading.

Table 10. Performance of SEntiMoji and Its Variants in Emotion Detection Tasks

Dataset	Class	Metric	SEntiMoji	SEntiMoji-G1	SEntiMoji-G2	SEntiMoji-T
JIRA-E1	Love	Precision	0.713	0.664	0.719	**0.743**
		Recall	0.674	**0.732**	0.554	0.692
		F1-score	0.692	0.694	0.626	**0.713**
		Accuracy	0.907	0.901	0.890	**0.915**
		Macro-precision	0.825	0.806	0.817	**0.843**
		Macro-recall	0.813	**0.832**	0.756	0.824
		Macro-F1	0.818	0.817	0.781	**0.831**
	Anger	Precision	0.835	0.658	0.704	**0.846**
		Recall	**0.664**	0.512	0.508	0.647
		F1-score	**0.739**	0.575	0.590	0.733
		Accuracy	**0.843**	0.744	0.761	0.841
		Macro-precision	0.840	0.717	0.742	**0.843**
		Macro-recall	**0.798**	0.687	0.699	0.793
		Macro-F1	**0.813**	0.696	0.711	0.809
	Sadness	Precision	0.931	0.900	0.741	**0.937**
		Recall	**0.765**	0.663	0.570	0.722
		F1-score	**0.838**	0.761	0.644	0.813
		Accuracy	**0.911**	0.875	0.810	0.900
		Macro-precision	**0.918**	0.884	0.786	0.914
		Macro-recall	**0.869**	0.815	0.742	0.849
		Macro-F1	**0.888**	0.838	0.757	0.873
SO-E	Love	Precision	0.807	0.788	0.763	**0.808**
		Recall	**0.814**	0.730	0.571	0.766
		F1-score	**0.810**	0.757	0.654	0.786
		Accuracy	**0.903**	0.881	0.846	0.894
		Macro-precision	**0.872**	0.849	0.814	0.865
		Macro-recall	**0.874**	0.831	0.755	0.852
		Macro-F1	**0.872**	0.839	0.777	0.858
	Joy	Precision	**0.744**	0.641	0.645	0.668
		Recall	**0.379**	0.152	0.100	0.262
		F1-score	**0.475**	0.243	0.173	0.374
		Accuracy	**0.913**	0.904	0.902	0.911
		Macro-precision	**0.832**	0.776	0.776	0.795
		Macro-recall	**0.629**	0.571	0.547	0.624
		Macro-F1	**0.675**	0.596	0.560	0.663
	Anger	Precision	**0.812**	0.721	0.709	0.801
		Recall	**0.677**	0.527	0.331	0.650
		F1-score	**0.738**	0.607	0.451	0.717
		Accuracy	**0.912**	0.875	0.852	0.906
		Macro-precision	**0.871**	0.810	0.787	0.863
		Macro-recall	**0.821**	0.740	0.65	0.807
		Macro-F1	**0.842**	0.766	0.683	0.830
	Sadness	Precision	0.794	0.690	0.723	**0.812**
		Recall	**0.455**	0.341	0.148	0.447
		F1-score	**0.577**	0.451	0.245	0.575
		Accuracy	0.968	0.960	0.956	**0.969**
		Macro-precision	0.884	0.829	0.841	**0.893**
		Macro-recall	**0.725**	0.666	0.572	0.721
		Macro-F1	**0.780**	0.715	0.612	0.779
JIRA-E2	Excitement	Precision	0.904	0.696	0.628	**0.909**
		Recall	0.942	0.716	0.625	**0.949**
		F1-score	0.922	0.704	0.627	**0.928**
	Relaxation	Precision	0.858	0.678	0.595	**0.872**
		Recall	**0.814**	0.442	0.454	0.804
		F1-score	0.834	0.529	0.515	**0.835**
	Stress	Precision	**0.853**	0.611	0.525	0.840
		Recall	0.721	0.444	0.337	**0.758**
		F1-score	0.780	0.513	0.411	**0.796**
	Depression	Precision	0.846	0.546	0.614	**0.866**
		Recall	0.841	0.416	0.512	**0.848**
		F1-score	0.842	0.471	0.558	**0.856**
	Neutral	Precision	0.911	0.668	0.646	**0.914**
		Recall	**0.965**	0.885	0.849	0.958
		F1-score	**0.937**	0.761	0.734	0.935
	Accuracy		0.886	0.653	0.622	**0.891**
	Macro-precision		0.875	0.640	0.602	**0.880**

(Continued.)

Table 10. Continued

Dataset	Class	Metric	SEntiMoji	SEntiMoji-G1	SEntiMoji-G2	SEntiMoji-T
	Macro-recall		0.856	0.581	0.555	**0.863**
	Macro-F1		0.863	0.596	0.569	**0.870**
Unified-E	Love	Precision	0.801	0.725	0.762	**0.806**
		Recall	**0.814**	0.765	0.588	0.791
		F1-score	**0.806**	0.743	0.664	0.797
		Accuracy	**0.907**	0.874	0.858	0.904
		Macro-precision	**0.871**	0.825	0.820	0.870
		Macro-recall	**0.875**	0.836	0.765	0.865
		Macro-F1	**0.872**	0.830	0.787	0.867
	Joy	Precision	**0.676**	0.395	0.615	0.670
		Recall	**0.346**	0.305	0.091	0.319
		F1-score	**0.457**	0.344	0.159	0.432
		Accuracy	**0.913**	0.877	0.898	0.911
		Macro-precision	**0.802**	0.658	0.759	0.797
		Macro-recall	**0.663**	0.625	0.542	0.650
		Macro-F1	**0.705**	0.638	0.552	0.692
	Anger	Precision	0.801	0.687	0.754	**0.806**
		Recall	**0.672**	0.643	0.378	0.666
		F1-score	**0.730**	0.663	0.503	0.728
		Accuracy	**0.896**	0.863	0.843	**0.896**
		Macro-precision	0.859	0.796	0.804	**0.860**
		Macro-recall	**0.814**	0.783	0.672	0.811
		Macro-F1	**0.833**	0.789	0.705	0.832
	Sadness	Precision	**0.864**	0.570	0.833	**0.864**
		Recall	0.605	0.598	0.412	**0.618**
		F1-score	0.711	0.582	0.551	**0.718**
		Accuracy	0.955	0.921	0.938	**0.956**
		Macro-precision	**0.913**	0.764	0.888	**0.913**
		Macro-recall	0.798	0.776	0.702	**0.804**
		Macro-F1	0.843	0.769	0.759	**0.847**

Note: For each metric, the highest value is highlighted with shading.

Analysis for sentiment detection: Table 12 presents the performance of SEntiMoji-T and its variants in sentiment detection tasks. We can observe that when we use 80% of the labeled data (i.e., T-80%), the performance is comparable with SEntiMoji-T. In terms of macro-F1, T-80% even obtains a better result than SEntiMoji-T on the Code Review dataset, although only with an increase of 0.011. On the Unified-S dataset, the macro-F1 scores obtained by SEntiMoji-T and T-80% are the same. On the remaining three datasets, T-80% performs slightly inferior than SEntiMoji-T, with an average decrease of 0.01. However, when we use 20% of the labeled data for training (i.e., T-20%), we can observe an obvious decrease in performance. On one hand, we can find that T-20% performs worse than SEntiMoji-T in almost all the metrics. On the other hand, in terms of macro-F1, T-20% has an average decrease of 0.056.

To intuitively understand the changes in performance when scaling down the labeled data, we then turn to the statistical results in Table 14 for further analysis. On any of the datasets for sentiment detection, we find that the performance difference between SEntiMoji-T and T-80% is not statistically significant, which is consistent with our finding that the performance of the two methods is comparable. On the JIRA and Code Review datasets, the performance becomes significantly worse until only 20% of the labeled data are remaining. However, on the Stack Overflow and Unified-S datasets, the performance is significantly worse even when 60% of the labeled data are remaining, and the performance gap gets larger with the reduction of the labeled data.

Analysis for emotion detection: Table 13 presents the performance of SEntiMoji-T and its variants in emotion detection tasks. We can observe that T-80% can obtain comparable performance with SEntiMoji-T, which is consistent with our finding in sentiment detection tasks. Specifically, in terms of macro-F1, SEntiMoji-T can outperform T-80% in five tasks, while T-80% can outperform SEntiMoji-T in seven tasks. In many tasks, even when we use only 20% of labeled data (i.e.,

Table 11. McNemar's Statistics between the Results of SEntiMoji and SEntiMoji-G1/G2/T on Each Dataset, with Adjusted p-values in Parentheses

	SEntiMoji-G1	SEntiMoji-G2	SEntiMoji-T
JIRA	**19.066 (0.000)**	**130.021 (0.000)**	5.134 (0.156)
Stack Overflow	**136.970 (0.000)**	**444.582 (0.000)**	0.004 (1.000)
Code Review	**44.981 (0.000)**	**76.257 (0.000)**	1.482 (1.000)
Java Library	**32.494 (0.000)**	**62.223 (0.000)**	1.266 (1.000)
Unified-S	**202.197 (0.000)**	**378.750 (0.000)**	0.431 (1.000)
JIRA-E1 (Love)	0.446 (1.000)	3.241 (0.448)	1.441 (1.000)
JIRA-E1 (Anger)	**46.782 (0.000)**	**36.285 (0.000)**	0.025 (1.000)
JIRA-E1 (Sadness)	**17.014 (0.000)**	**68.966 (0.000)**	3.030 (0.497)
SO-E (Love)	**29.312 (0.000)**	**112.953 (0.000)**	**9.333 (0.014)**
SO-E (Joy)	**17.794 (0.000)**	**25.442 (0.000)**	4.124 (0.277)
SO-E (Anger)	**70.240 (0.000)**	**146.616 (0.000)**	4.050 (0.282)
SO-E (Sadness)	**19.557 (0.000)**	**29.867 (0.000)**	0.000 (1.000)
JIRA-E2	**325.635 (0.000)**	**388.418 (0.000)**	0.598 (1.000)
Unified-E (Love)	**87.695 (0.000)**	**111.573 (0.000)**	1.053 (1.000)
Unified-E (Joy)	**128.380 (0.000)**	**27.458 (0.000)**	0.709 (1.000)
Unified-E (Anger)	**87.862 (0.000)**	**126.185 (0.000)**	0.005 (1.000)
Unified-E (Sadness)	**144.966 (0.000)**	**46.312 (0.000)**	0.136 (1.000)

Note: Results significant at 5% level are highlighted with shading.

T-20%), the performance is still comparable with SEntiMoji-T. For example, in the sadness detection task of the JIRA-E1 dataset, the macro-F1 scores obtained by SEntiMoji-T and T-20% are 0.873 and 0.872, with only a trivial gap of 0.001. However, in some tasks, using such a small amount of labeled data can result in an obvious decrease in performance. For instance, in the anger detection task in JIRA-E1, the macro-F1 of T-20% is 0.087 lower than the one obtained by SEntiMoji-T.

Then we analyze the McNemar's statistics between the results of SEntiMoji-T and its variants in Table 14. In 11 of 12 emotion tasks, the performance maintains at a comparable when 60% of the labeled data are excluded (i.e., T-40%). In addition, in five tasks, the performance is still comparable even when 80% of labeled data are excluded (i.e., T-20%).

Based on the above analysis for sentiment and emotion detection, we can find that a certain amount of labeled data are essential in some tasks such as sentiment detection on Stack Overflow and Unified-S datasets. However, in most tasks, the combination of the large-scale Tweets and not so many labeled technical texts is able to obtain satisfying results, which demonstrates the importance of the general sentimental knowledge incorporated by the Twitter data.

Ans. to RQ3: The large-scale Twitter data rather than the GitHub data make a key contribution to the power of SEntiMoji. Although a certain amount of labeled data are essential in some tasks, the combination of Twitter data and not so many labeled technical texts is able to obtain satisfying results in most tasks.

5.4 RQ4: What Are the Main Difficulties Faced by SEntiMoji?

Although we have investigated SEntiMoji's "secret of success," we are also curious about the situations where SEntiMoji yields a wrong classification. Specifically, we aim to manually examine the misclassified cases and analyze the error causes, by which we can distill the difficulties faced by SEntiMoji and identify open challenges of sentiment and emotion detection in SE.

Table 12. Performance of SEntiMoji-T and Its Variants in Sentiment Detection Tasks

Dataset	Class	Metric	SEntiMoji-T	T-80%	T-60%	T-40%	T-20%
JIRA	Pos	Precision	0.939	**0.952**	0.948	0.940	0.950
		Recall	0.933	0.931	0.935	**0.937**	0.921
		F1-score	0.936	**0.941**	**0.941**	0.938	0.935
	Neu	Precision	**0.808**	0.795	0.798	0.779	0.775
		Recall	0.866	**0.887**	0.883	0.861	0.861
		F1-score	**0.845**	0.838	0.838	0.817	0.815
	Neg	Precision	**0.913**	0.905	0.905	0.901	0.879
		Recall	**0.839**	0.835	0.835	0.813	0.825
		F1-score	**0.874**	0.868	0.868	0.854	0.851
	Accuracy		**0.893**	0.891	0.891	0.880	0.876
	Macro-precision		**0.887**	0.884	0.884	0.874	0.868
	Macro-recall		**0.886**	0.884	0.884	0.870	0.869
	Macro-F1		**0.885**	0.883	0.883	0.870	0.867
Stack Overflow	Pos	Precision	0.921	**0.926**	0.922	0.923	0.918
		Recall	**0.937**	0.929	0.930	0.922	0.912
		F1-score	**0.929**	0.928	0.926	0.926	0.915
	Neu	Precision	**0.841**	0.830	0.826	0.819	0.813
		Recall	**0.840**	0.837	0.830	0.836	0.828
		F1-score	**0.841**	0.834	0.828	0.827	0.820
	Neg	Precision	**0.854**	0.840	0.837	0.834	0.825
		Recall	**0.838**	0.827	0.827	0.810	0.811
		F1-score	**0.845**	0.833	0.828	0.828	0.817
	Accuracy		**0.873**	0.866	0.862	0.858	0.852
	Macro-precision		**0.872**	0.866	0.862	0.859	0.852
	Macro-recall		**0.872**	0.865	0.860	0.856	0.850
	Macro-F1		**0.872**	0.866	0.861	0.857	0.851
Code Review	Non-Neg	Precision	0.850	**0.857**	**0.857**	0.839	0.818
		Recall	**0.958**	0.950	0.948	0.957	0.948
		F1-score	**0.900**	**0.900**	0.899	0.894	0.877
	Neg	Precision	**0.793**	0.778	0.774	0.778	0.717
		Recall	0.491	0.521	**0.527**	0.447	0.364
		F1-score	0.603	0.614	**0.618**	0.561	0.469
	Accuracy		**0.841**	**0.841**	**0.841**	0.829	0.801
	Macro-precision		**0.821**	0.815	0.816	0.809	0.767
	Macro-recall		0.724	**0.741**	0.737	0.702	0.656
	Macro-F1		0.752	**0.763**	0.759	0.727	0.673
Java Library	Pos	Precision	**0.878**	0.825	0.803	0.828	0.728
		Recall	**0.281**	0.214	0.222	0.192	0.142
		F1-score	**0.426**	0.337	0.343	0.309	0.235
	Neu	Precision	**0.869**	0.863	0.862	0.850	0.836
		Recall	0.971	0.967	0.972	0.972	**0.979**
		F1-score	**0.917**	0.912	0.914	0.907	0.902
	Neg	Precision	0.734	0.737	0.735	0.706	**0.769**
		Recall	0.513	**0.518**	0.490	0.412	0.335
		F1-score	**0.599**	0.598	0.586	0.513	0.448
	Accuracy		**0.859**	0.849	0.849	0.838	0.829
	Macro-precision		**0.827**	0.800	0.800	0.795	0.778
	Macro-recall		**0.589**	0.576	0.562	0.525	0.486
	Macro-F1		**0.647**	0.626	0.614	0.576	0.528
Unified-S	Pos	Precision	0.914	**0.922**	0.907	0.892	0.877
		Recall	**0.897**	0.895	0.889	0.862	0.852
		F1-score	0.906	**0.908**	0.898	0.877	0.864
	Neu	Precision	**0.829**	0.823	0.813	0.786	0.775
		Recall	0.865	**0.875**	0.858	0.831	0.816
		F1-score	0.846	**0.848**	0.835	0.808	0.795
	Neg	Precision	0.843	**0.848**	0.835	0.810	0.805
		Recall	**0.801**	0.788	0.780	0.767	0.761
		F1-score	**0.820**	0.816	0.806	0.787	0.781
	Accuracy		0.859	**0.860**	0.848	0.825	0.814
	Macro-precision		0.862	**0.864**	0.852	0.829	0.819
	Macro-recall		**0.854**	0.853	0.842	0.820	0.810
	Macro-F1		**0.857**	**0.857**	0.846	0.824	0.813

Note: For each metric, the highest value is highlighted with shading.

Table 13. Performance of SEntiMoji and Its Variants in Emotion Detection Tasks

Dataset	Class	Metric	SEntiMoji-T	T-80%	T-60%	T-40%	T-20%
JIRA-E1	Love	Precision	**0.743**	0.679	0.646	0.627	0.642
		Recall	0.692	0.638	0.614	0.567	0.455
		F1-score	**0.713**	0.652	0.626	0.593	0.526
		Accuracy	**0.915**	**0.915**	0.908	0.903	0.896
		Macro-precision	**0.843**	0.816	0.806	0.789	0.784
		Macro-recall	**0.824**	0.792	0.789	0.760	0.723
		Macro-F1	**0.831**	0.800	0.796	0.772	0.744
	Anger	Precision	**0.846**	0.822	0.798	0.808	0.817
		Recall	0.647	**0.657**	0.637	0.642	0.551
		F1-score	**0.733**	0.729	0.708	0.711	0.647
		Accuracy	**0.841**	0.838	0.825	0.830	0.801
		Macro-precision	**0.843**	0.839	0.824	0.828	0.814
		Macro-recall	0.793	**0.797**	0.784	0.786	0.746
		Macro-F1	0.809	**0.811**	0.798	0.799	0.761
	Sadness	Precision	0.937	0.927	**0.951**	0.902	0.933
		Recall	0.722	**0.758**	0.728	0.728	0.721
		F1-score	0.813	**0.833**	0.823	0.805	0.813
		Accuracy	0.900	**0.908**	0.906	0.894	0.900
		Macro-precision	**0.914**	0.910	**0.914**	0.896	0.909
		Macro-recall	**0.849**	0.861	**0.849**	0.848	**0.849**
		Macro-F1	0.873	**0.880**	0.872	0.867	0.872
SO-E	Love	Precision	**0.808**	0.785	0.779	0.771	0.769
		Recall	0.766	**0.796**	0.789	0.779	0.750
		F1-score	0.786	**0.790**	0.784	0.774	0.759
		Accuracy	**0.894**	0.893	0.889	0.885	0.880
		Macro-precision	**0.865**	0.863	0.860	0.855	0.851
		Macro-recall	0.852	**0.863**	0.860	0.854	0.842
		Macro-F1	0.858	**0.863**	0.860	0.854	0.846
	Joy	Precision	**0.668**	0.666	0.624	0.577	0.567
		Recall	0.262	**0.288**	0.213	0.201	0.123
		F1-score	0.374	**0.399**	0.314	0.296	0.198
		Accuracy	0.911	**0.913**	0.908	0.906	0.901
		Macro-precision	**0.795**	0.806	0.788	0.766	0.760
		Macro-recall	0.624	**0.638**	0.606	0.595	0.566
		Macro-F1	0.663	**0.681**	0.642	0.627	0.588
	Anger	Precision	0.801	**0.815**	0.801	0.802	0.808
		Recall	0.650	**0.661**	**0.661**	0.637	0.593
		F1-score	0.717	**0.729**	0.724	0.708	0.682
		Accuracy	0.906	**0.910**	0.908	0.904	0.899
		Macro-precision	0.863	**0.870**	0.865	0.863	0.862
		Macro-recall	0.807	**0.818**	0.816	0.804	0.787
		Macro-F1	0.830	**0.840**	0.838	0.828	0.816
	Sadness	Precision	0.812	0.784	0.837	0.831	**0.867**
		Recall	**0.447**	0.442	0.405	0.411	0.333
		F1-score	**0.575**	0.564	0.542	0.548	0.473
		Accuracy	**0.969**	0.967	0.967	0.968	0.965
		Macro-precision	0.893	0.872	0.893	0.900	**0.910**
		Macro-recall	**0.721**	0.718	0.700	0.705	0.665
		Macro-F1	**0.779**	0.772	0.760	0.767	0.727
JIRA-E2	Excitement	Precision	**0.909**	**0.909**	**0.909**	0.900	0.859
		Recall	0.949	0.943	0.940	0.943	**0.951**
		F1-score	**0.928**	0.925	0.924	0.921	0.902
	Relaxation	Precision	0.872	0.859	0.847	0.844	**0.879**
		Recall	0.804	0.803	**0.821**	0.771	0.690
		F1-score	**0.835**	0.829	0.831	0.804	0.772
	Stress	Precision	0.840	**0.848**	0.771	0.810	0.719
		Recall	**0.758**	0.736	0.696	0.664	0.652
		F1-score	**0.796**	0.787	0.728	0.726	0.679
	Depression	Precision	**0.866**	0.856	0.831	0.813	0.794
		Recall	**0.848**	0.842	0.795	0.822	0.702
		F1-score	**0.856**	0.848	0.812	0.817	0.744
	Neutral	Precision	**0.914**	0.912	0.913	0.901	0.867
		Recall	0.958	**0.966**	0.950	0.956	0.947
		F1-score	0.935	**0.938**	0.931	0.928	0.905
	Accuracy		**0.891**	0.888	0.871	0.869	0.836
	Macro-precision		**0.880**	0.877	0.854	0.854	0.824

(Continued.)

Table 13. Continued

Dataset	Class	Metric	SEntiMoji-T	T-80%	T-60%	T-40%	T-20%
	Macro-recall		**0.863**	0.858	0.840	0.831	0.789
	Macro-F1		**0.870**	0.865	0.845	0.839	0.800
Unified-E	Love	Precision	**0.806**	0.790	0.805	0.795	0.781
		Recall	**0.791**	0.768	0.774	0.765	0.715
		F1-score	**0.797**	0.778	0.788	0.779	0.745
		Accuracy	**0.904**	0.896	0.901	0.897	0.883
		Macro-precision	**0.870**	0.859	0.868	0.861	0.847
		Macro-recall	**0.865**	0.852	0.857	0.851	0.826
		Macro-F1	**0.867**	0.855	0.862	0.856	0.835
	Joy	Precision	0.670	**0.674**	0.646	0.629	0.558
		Recall	**0.319**	0.299	0.276	0.294	0.237
		F1-score	**0.432**	0.414	0.386	0.401	0.323
		Accuracy	**0.911**	**0.911**	0.907	0.908	0.898
		Macro-precision	0.797	0.798	**0.783**	0.775	0.737
		Macro-recall	**0.650**	0.641	0.629	0.637	0.607
		Macro-F1	**0.692**	0.683	0.668	0.675	0.634
	Anger	Precision	**0.806**	0.796	0.789	0.778	0.751
		Recall	0.666	**0.699**	0.666	0.667	0.643
		F1-score	0.728	**0.743**	0.722	0.715	0.691
		Accuracy	0.896	**0.899**	0.892	0.889	0.880
		Macro-precision	**0.860**	0.859	0.852	0.846	0.830
		Macro-recall	0.811	**0.826**	0.809	0.808	0.793
		Macro-F1	0.832	**0.840**	0.827	0.823	0.808
	Sadness	Precision	0.864	0.891	0.886	**0.893**	0.883
		Recall	**0.618**	0.614	0.588	0.566	0.533
		F1-score	0.718	**0.723**	0.705	0.690	0.663
		Accuracy	0.956	**0.957**	0.955	0.954	0.951
		Macro-precision	0.913	**0.926**	0.923	0.925	0.919
		Macro-recall	**0.804**	0.803	0.790	0.779	0.763
		Macro-F1	0.847	**0.850**	0.840	0.833	0.818

Note: For each metric, the highest value is highlighted with shading.

5.4.1 Error Analysis of Sentiment Detection. On JIRA, Stack Overflow, Code Review, and Java Library datasets, SEntiMoji misclassifies 452 samples in total. To achieve 95% confidence level and 5% confidence interval, we randomly select 208 misclassified samples for manual inspection. We follow an open coding method [67] to inspect and classify selected samples into different error causes. First, the first two authors jointly inspect 50 samples to distill the initial set of error causes and develop a coding guide with definitions and examples for each identified cause. Then, the two authors use the coding guide to independently label the remaining 158 samples, each of which is annotated with the most possible error cause. To measure the inter-rater agreement, we employ *Cohen' Kappa* (κ) [19] and have $\kappa = 0.819$. Such a statistic indicates almost perfect agreement [62] and evidences the reliability of the coding schema and procedure. Finally, the samples where two authors assign different labels are discussed to determine their final labels. To resolve conflicts, a colleague, who is familiar with our research, is introduced as an arbitrator [21].

In total, we distill seven error causes of sentiment detection. The distribution of these categories is shown in Table 15. Each category is described as follows.

C1: Implicit sentiment. About 31.7% of errors are due to the implicit sentiment polarity. In many cases, developers use neutral expressions to convey sentiment. Such sentiment can be captured by annotators due to their rich knowledge, but it is difficult for a machine learning model whose knowledge is only from the training data. For example, the sample "My patch wouldn't compile" (id: 937006_1, JIRA dataset) conveys a negative sentiment, but it contains no negative terms. Therefore, it is too obscure for SEntiMoji to classify it correctly.

C2: Misinterpreted purposes. This category accounts for 25.0% of bad cases. It includes neutral requests, questions, and tips that are mistaken as positive or negative. For example, "Avoid legacy date-time classes" (id: 5122, Java Library dataset) is considered by SEntiMoji as negative, although it is just a tip. In addition, this category also includes objective polar utterances, which

Table 14. McNemar's Statistics between the Results of SEntiMoji-T and Its Variants on Each Dataset, with Adjusted p-values in Parentheses

	T-80%	T-60%	T-40%	T-20%
JIRA	0.114 (1.000)	0.078 (1.000)	7.585 (0.093)	**9.551 (0.034)**
Stack Overflow	4.571 (0.399)	**10.527 (0.019)**	**16.962 (0.000)**	**32.761 (0.000)**
Code Review	0.000 (1.000)	0.010 (1.000)	2.676 (1.000)	**29.184 (0.000)**
Java Library	2.086 (1.000)	1.639 (1.000)	**9.592 (0.034)**	**17.204 (0.000)**
Unified-S	0.056 (1.000)	**29.698 (0.000)**	**270.951 (0.000)**	**377.065(0.000)**
JIRA-E1 (Love)	1.441 (1.000)	2.065 (1.000)	7.220 (0.104)	**12.444 (0.000)**
JIRA-E1 (Anger)	0.000 (1.000)	0.468 (1.000)	0.721 (1.000)	**11.223 (0.019)**
JIRA-E1 (Sadness)	0.028 (1.000)	0.000 (1.000)	0.356 (1.000)	0.075 (1.000)
SO-E (Love)	1.225 (1.000)	0.473 (1.000)	0.293 (1.000)	2.070 (1.000)
SO-E (Joy)	1.363 (1.000)	1.152 (1.000)	2.064 (1.000)	8.347 (0.065)
SO-E (Anger)	1.900 (1.000)	0.255 (1.000)	0.318 (1.000)	3.971 (0.518)
SO-E (Sadness)	2.485 (1.000)	1.639 (1.000)	0.742 (1.000)	4.438 (0.408)
JIRA-E2	0.155 (1.000)	6.781 (0.128)	**11.236 (0.019)**	**43.584 (0.000)**
Unified-E (Love)	**14.135 (0.000)**	1.092 (1.000)	6.695 (0.136)	**41.564 (0.000)**
Unified-E (Joy)	0.102 (1.000)	5.878 (0.196)	3.341 (0.740)	**24.790 (0.000)**
Unified-E (Anger)	1.964 (1.000)	1.842 (1.000)	5.659 (0.214)	**23.615 (0.000)**
Unified-E (Sadness)	1.167 (1.000)	0.161 (1.000)	1.779 (1.000)	**10.112 (0.019)**

Note: Results significant at 5% level are highlighted with shading.

Table 15. The Distribution of Error Categories in Sentiment Detection

Category	# of samples	Proportion
C1: Implicit sentiment	66	31.7%
C2: Misinterpreted purposes	52	25.0%
C3: Complex context information	38	18.3%
C4: Failure to capture obvious sentiment	34	16.3%
C5: Politeness	13	6.3%
C6: Data pre-processing	3	1.4%
C7: Figurative language	2	1.0%

describe positive or negative factual information about something without conveying a private state [109]. Take the post "TestBlockTokenWithDFS fails for the same reason" as an example. It reports a problem that may cause negative results and thus is classified as negative by SEntiMoji. However, it is an objective polar utterance that does not convey any emotion or opinion.

C3: Complex context information. About 18.3% of bad cases can be attributed to complex context information. On one hand, some samples are compound sentences that contain some conjunction words such as "but," "and," "unless," and "so." In such samples, sub-sentences may convey different sentiment, which makes it challenging for a machine learning model to determine the overall sentiment. For example, in the negative sample "Also your username was stupidly named and confusing, so I deleted it and created a new one more appropriate" (id: 2148299_4, JIRA dataset), the first sub-sentence conveys obviously negative sentiment, but the second sub-sentence is neutral. As a result, SEntiMoji misclassifies it as neutral. In addition, some samples are document-level, i.e., a sample may contain more than one sentence. However, all the existing sentiment detection methods (including SEntiMoji) consider the entire content of a sample as the unit of analysis, regardless whether the sample is a sentence or a document. Under such a situation, document-level samples consisting of sentences conveying different sentiment make the sentiment detection task challenging.

Table 16. The Distribution of Error Categories in Emotion Detection

Category	# of samples	Proportion
C1: Implicit emotion	34	10.7%
C2: Misinterpreted purposes	12	3.8%
C3: Complex context information	98	30.7%
C4: Failure to capture obvious emotion	92	28.8%
C5: Politeness	5	1.6%
C6: Data pre-processing	2	0.6%
C7: Figurative language	1	0.3%
C8: The same emotion polarity	75	23.5%

C4: Failure to capture obvious sentiment. Although SEntiMoji can outperform existing methods in SE, it still cannot correctly classify some samples with obvious sentiment. For example, SEntiMoji considers "Seems to work reliably now" (id: 719536_1, JIRA dataset) neutral, but in fact it conveys an obviously positive sentiment. Cases attributed to this category account for about 16.3%.

C5: Politeness. In online communication, developers may use lexical cues of politeness, such as "please," "thanks," "sorry," and "I'm afraid," to decorate a neutral description. For example, the sample "Thanks Sian. Patch applied in SECURITY module and third party notices file at repo revision r543463. Please check it was applied as you expected" (id: 1050218, JIRA dataset) expresses no sentiment, but the polite cues (i.e., "Thanks" and "please") mislead SEntiMoji to consider it positive. This category accounts for about 6.3% of bad cases.

C6: Data pre-processing. SEntiMoji needs to pre-process texts before learning representations of them, to filter out some noise. However, data pre-processing can also filter out some useful information. For instance, in the sample "+1 on intent from looking at what the patch fixes" (id: 819316_1, JIRA dataset), "+1" conveys positive sentiment. However, it is replaced with a specific token during data pre-processing, which makes this sample be classified as neutral. The bad cases in this category account for only 1.4%.

C7: Figurative language. Figurative language can be used to express metaphor, sarcasm, irony, humor, and so on. The usage of it can make sentiment detection challenging. For example, "Regex is your friend" (id: 695065_2, JIRA dataset) indicates a positive attitude to regex, but such sentiment is implied in the metaphor and thus cannot be captured by SEntiMoji. Another sample we identified in this category is "Well, the funny part is, we do not know" (id: 4578, Java Library dataset), which conveys negative sentiment through an expression of irony but is classified by SEntiMoji as neutral.

5.4.2 Error Analysis for Emotion Detection. As is described before, for emotion detection, we perform binary classifications on the JIRA-E1 and SO-E datasets. For example, in the love task of JIRA-E1, we classify each sample as love or no-love. In the ten binary classification tasks, 1,863 bad cases are generated. To achieve 95% confidence level and 5% confidence interval, we randomly select 319 bad cases for manual inspection. Similar to the error analysis for sentiment detection, we follow an open coding method to distill categories of error causes and then label each sampled case with one category. During the independent labeling process, the inter-rater agreement is almost perfect given $k = 0.858$. Finally, we obtain eight categories and their distribution is summarized in Table 16.

The first seven categories are almost the same as those obtained in the error analysis for sentiment detection. Take the first category (C1) as an example. Just like that implicit sentiment can

Table 17. The Distribution of Misclassified Cases Obtained by SEntiMoji on JIRA-E2

		Predicted Emotion				
		Excitement	Relaxation	Stress	Depression	Neutral
Ground Truth	Excitement	-	**15**	3	2	4
	Relaxation	**22**	-	2	3	15
	Stress	12	7	-	**32**	20
	Depression	5	4	18	-	**19**
	Neutral	2	5	**8**	7	-

Note: The highest value of each row is highlighted with shading.

mislead SEntiMoji to classify sentimental samples as neutral, implicit emotion can also mislead SEntiMoji to classify emotional samples as absence of emotion. To save space, here, we do not describe the remaining six categories one by one. What we want to mention is the distribution of the first seven categories. In sentiment detection, the seven categories in descending order of their frequencies are C1, C2, C3, C4, C5, C6, and C7. However, in emotion detection, C3 and C4 occur much more frequently than in sentiment detection. This phenomenon can be explained as follows:

- *Explanation for C3*: In each of the six classification tasks on SO-E, the entire 4,800 document-level samples are used for evaluation. Considering that JIRA-E1 has only 4,000 sentence-level samples, with 1,000 samples per classification task, we can find that a large proportion of samples used for the binary emotion detection are documents rather than sentences. Since SEntiMoji considers the entire content of each sample (regardless whether it is a sentence or a document) as the unit of analysis, these document-level samples have a high chance of being misclassified due to the complex context information (C3) in the documents.
- *Explanation for C4*: Compared to sentiment detection, emotion detection tends to have a more imbalanced data distribution, as samples conveying a specific emotion are usually only a subset of positive or negative samples. In other words, emotion detection is more challenging than sentiment detection. Take the sadness task of SO-E as an example. As there are only 4.8% of the samples conveying sadness, SEntiMoji has only a little knowledge about such an emotion and thus may misclassify some obviously sad samples as no-sadness. As a result, we find that C4 accounts for such a large proportion.

Besides the seven categories shared by sentiment and emotion detection, we find a new added category for emotion detection:

C8: The same emotion polarity. Some emotions, especially those with the same polarity, are relatively similar. For instance, joy and love are both positive emotions, so it is difficult for a machine learning model to distinguish them. Take the post "Great point - will change!" (id: t2837, SO-E dataset) as an example. SEntiMoji classifies it as love due to the positive emotion contained in it, but actually it does not convey the love emotion. We find that 23.5% of bad cases can be attributed to this category.

The severity of C8 can be also evidenced by the distribution of the cases misclassified by SEntiMoji on JIRA-E2 dataset (see Table 17). As excitement and relaxation are both positive, the most common mistakes in the two emotions are to classify them as each other. Similarly, stress samples are often identified as depression.

Ans. to RQ4: For sentiment detection, SEntiMoji mainly faces challenges caused by implicit sentiment, misinterpreted purposes, complex context information, failure to capture obvious sentiment, politeness, data pre-processing, and figurative language. For emotion detection, besides these challenges, SEntiMoji also faces the challenge caused by the same emotion polarity.

Table 18. Confusion Matrices Obtained by SEntiMoji on Java Library Dataset

		Predicted Polarity		
		Positive	Neutral	Negative
Ground Truth	Positive	43	84	4
	Neutral	6	1,148	37
	Negative	2	72	104

Table 19. Confusion Matrices Obtained by SentiCR on the Samples in Java Library Dataset That Are Misclassified by SentiCR But Correctly Classified by SEntiMoji

		Predicted Polarity		
		Positive	Neutral	Negative
Ground Truth	Positive	0	12	2
	Neutral	28	0	54
	Negative	2	27	0

6 LESSONS LEARNED AND IMPLICATIONS

In this section, we summarize the lessons learned from our study and try to propose some implications for future research.

- **SEntiMoji can benefit some tasks that rely on identifying sentiment and emotion from SE-related texts.** Results in Section 5 demonstrate that SEntiMoji can improve some sentiment- and emotion-enabled tasks in SE. Take Java Library dataset as an example. The original intention of it is to mine developers' opinions toward different software libraries, which can facilitate library recommendation in the software development [61]. In terms of the performance of SentiCR on this dataset, nearly half of the samples classified as positive or negative are actually false (precision@pos: 0.553, precision@neg: 0.546), which is far from being qualified for evaluating a software library. The precision levels of other existing methods are even lower. By comparison, SEntiMoji can achieve a comparable recall level but significantly improve the precision over positive and negative samples (precision@pos: 0.849, precision@neg: 0.729). We further inspect its confusion matrices on Java Library dataset (see Table 18), and find that bad cases mainly focus on classifying positive and negative samples as neutral rather than as the opposite sentiment polarity. The results indicate that although SEntiMoji may miss some sentimental posts, software libraries predicted by SEntiMoji to be very positively or negatively discussed are credible, which meets the criterion in such recommendation tasks that finding the right candidates is usually far more important than finding as many candidates as possible [108]. To further demonstrate the improvement achieved by SEntiMoji, we then focus on the samples where SentiCR yields a wrong prediction but SEntiMoji makes a correct prediction. In Java Library dataset, there are 125 such samples. The confusion matrices obtained by SentiCR on these samples are presented in Table 19. We can find that most errors concentrate on misclassifying neutral samples as positive or negative, which makes the sentiment detected by SentiCR toward different software libraries unreliable. Fortunately, these neutral samples can be correctly classified by SEntiMoji.

In other tasks such as sentiment/emotion detection to improve developers' productivity, SEntiMoji is also potentially helpful. When a project manager finds that developers are suffering from negative sentiment or emotion reported by SEntiMoji, it is true with a high probability and worth paying attention to. Therefore, the manager can take necessary actions to defuse the situation.

Additionally, SEntiMoji can benefit the empirical studies that investigate the correlation between developers' sentiment/emotion and other SE-related factors. For example, when exploring the correlation between negative sentiment and issue fixing time [81], the conclusion might be biased if many identified negative posts are not really negative. The high precision of SEntiMoji on sentimental and emotional posts can alleviate this bias to some extent.

Although SEntiMoji achieves satisfactory precision in comparison with previous methods, we should admit that more efforts should be devoted to further improving its recall level. If the recall level is further improved, then SEntiMoji can be more qualified for the aforementioned tasks in SE. For example, with a higher recall level, SEntiMoji can be more sensitive to negative emotions of developers and thus inform the project managers more promptly.

Finally, we urge researchers and practitioners to select the appropriate method in line with the intended goals. Before selecting the method, they should first make sure which sentiment/emotion is the focus of the application scenario. For example, if project managers want to detect whether developers feel stressed over a period of time, then stress is the focus of this task. Given the satisfying performance of SEntiMoji in detecting stress, SEntiMoji can be appropriate. However, if the managers want to know if developers lack courage or feel fearful on new tasks, they can choose other methods, since SEntiMoji cannot outperform existing methods in detecting fear. In addition, the granularity of emotion detection also depends on the application scenarios. In the applications just described, we only need a simple binary classifier to detect the presence of stress/fear. However, in some scenarios, a fine-grained emotion classifier is needed. For example, Ortu et al. [81] aimed to investigate the relation between different emotions (e.g., love, joy, anger, and sadness) with the time to fix a JIRA issue, so they need a fine-grained classifier that can identify all these emotions.

- **Researchers can try to leverage the general affective expressions from the open domain via transfer learning, rather than unilaterally pursuing the domain-specific knowledge from the limited labeled data in SE.** Previous studies found that sentiment detection tools trained on non-technical texts are not always adequate for SE tasks [48, 54] and a lack of technical knowledge is the main reason [48]. Since then, many studies focused on how to use labeled SE-related texts to train SE-customized sentiment and emotion classifiers [1, 10, 12, 48, 50]. However, in fact, there are many general affective expressions shared by SE and open domains (e.g., social media). Therefore, we are interested that whether the potential value of such information has been deeply explored in previous studies. Since the customization of sentiment and emotion detection for SE is still at dawn, the labeled SE-related texts are too scarce to train a high-quality classification model. Therefore, we should not unilaterally place emphasis on learning the technical knowledge based on existing already-labeled SE-related data. In this study, we demonstrate that leveraging the general affective information from large-scale social media data via transfer learning can exactly facilitate sentiment and emotion detection in SE.
- **The construction process of datasets can affect the evaluation, so the performance on different datasets should be analyzed more rationally.** When comparing SEntiMoji with existing baseline methods in sentiment detection tasks, we find that performance gaps vary a lot on different datasets. It is not the first time that such a phenomenon is observed and reported. Novielli et al. [79] attributed this result to the different labeling approaches (i.e., model-driven annotation or ad hoc annotation). In this study, we turn to the construction process of these datasets to seek the reason. For example, the Stack Overflow dataset is constructed based on the corpus filtered by SentiStrength. Therefore, when comparing the performance of SentiStrength and other methods on this dataset, we should be careful to avoid being biased.

In addition, both the Stack Overflow and Code Review datasets are pre-processed to avoid imbalanced class distribution. It makes the classification task much easier and may not match their

target application scenarios. If researchers want to apply a sentiment detection method to a technical Q&A site such as Stack Overflow where the sentiment distribution is highly imbalanced, then they should not be blinded by the satisfactory results obtained on such datasets. Of course, we cannot deny or ignore that in some applications, the distribution of sentiment is relatively balanced. For these applications, evaluations on such processed datasets are meaningful.

- **Researchers can put more attention on the identified serious error causes, such as complex context information and implicit sentiment/emotion, to further improve the performance of SE-customized sentiment and emotion detection.** Our error analysis in Section 5.4 identifies open challenges in sentiment and emotion detection in SE, which can facilitate the future work. Researchers can put more attention on the dominant error causes. Techniques in other communities such NLP can be borrowed to alleviate these problems. Take *C3: Complex context information* as an example. SE-related texts often contain more than one sentence and the different semantics of sentences in a text make sentiment and emotion identification tasks difficult. In NLP, some research efforts have been devoted to sentiment detection in document level [27, 112]. Researchers can borrow some insights from these studies to improve the sentiment and emotion detection tasks in SE.
- **The characteristics of datasets can affect the results of error analysis, so the severity of different error causes should be considered rationally.** When comparing the distribution of error causes in sentiment and emotion detection, we find some inconsistencies. For example, because emotion detection is mainly performed on documents rather than sentences, we find that complex context information is most severe in this task. By comparison, less document-level samples are used in the evaluation of sentiment detection, so such a problem is covered in the error analysis for sentiment detection to some extent. This finding informs researchers to consider the severity of different error causes rationally. If researchers want to apply the sentiment and emotion detection methods to document-level texts, such as issue comments and Stack Overflow posts, then *C3: complex context information* should be taken seriously. However, if the texts in their application scenarios are mainly sentences, this error cause can be not such malicious.

7 THREATS TO VALIDITY

Threats to construct validity concern the relation between theory and observation. The JIRA dataset is originally labeled with different emotions. To use this dataset for sentiment detection, we follow previous studies to map the multi-class emotions to trinary sentiment polarity labels and filter some ambiguous samples. This process may violate the original distribution of this dataset and lower the difficulty of the classification task, which can affect the performance of different methods. However, fortunately, we have several benchmark datasets for a comprehensive comparison and the superiority of SEntiMoji is not only observed on the JIRA dataset.

Threats to internal validity concern confounding factors that could affect the obtained results. In our study, they are mainly from the configuration of baseline methods. We replicate these methods by using their released scripts or recommended hyper-parameter settings. However, some hyper-parameters can be further tuned to improve the classification performance.

In addition, SEntiMoji also has space for further improvement. To fine-tune the pre-trained DeepMoji, we select only those GitHub samples containing any of the 64 emojis predicted by DeepMoji. Limiting the choice to only those 64 emojis may discard other emojis that appear more frequently in GitHub posts and in result hinder our model to capture the affective expressions expressed in GitHub posts. Although SEntiMoji has already outperformed existing methods in most cases, we still encourage future work to further capture the affective information contained in GitHub posts and improve the performance.

What is more, we observe that SEntiMoji cannot significantly outperform existing methods in the surprise and fear tasks where data distribution is very imbalanced. Since data sampling techniques (e.g., over-sampling) are demonstrated to be beneficial to tackle the imbalanced data distribution, integrating such techniques into SEntiMoji may further improve the performance.

Furthermore, we find that 6.3% of misclassified samples in sentiment detection can be attributed to the usage of lexical cues of politeness in neutral descriptions. Therefore, politeness detection [20] may be useful for improving SEntiMoji. However, note that since lexical cues of politeness can also be used in positive or negative texts, it is impractical to directly use such approaches to filter out samples expressing politeness. Further research efforts are needed to adapt these approaches to improve existing sentiment detection methods.

Moreover, manual error analysis presents the threat of subjectivity. To minimize this threat, two authors individually inspect the bad cases and finally reach agreement through discussions. The inter-rater agreement is relatively high, which ensures the reliability of the coding schema and procedure. Furthermore, the conflicts are resolved by introducing an experienced colleague.

Threats to external validity concern the generalizability of our experimental results. Our evaluation has covered several mostly used benchmark datasets. However, we still cannot claim that the performance of SEntiMoji can be generalized across datasets, because the selected datasets cannot represent all types of texts in SE. In addition, various sentiment and emotion detection methods have been used in SE. In our study, we select only some representative ones for comparison.

What is more, to tackle the multi-label classification task on SO-E dataset, we follow the previous work [12] to employ the binary relevance method, i.e., transforming it into binary classification tasks. Besides this common method, another way is to transform it into a multi-class classification task [96], where each class represents a label combination. For example, if we have a multi-label classification task that involves two classes (i.e., A and B), we can transform the tasks into a multi-class classification that involves four classes, i.e., those labeled with A but not B, those labeled with B but not A, those labeled with both A and B, and those not labeled with A or B. However, in this article, we evaluate SEntiMoji on multi-label classification only using the binary relevance method.

8 RELATED WORK

We then describe the literature related to this study. Our research is particularly inspired by two streams of literature, i.e., sentiment and emotion detection in SE and emojis in sentiment and emotion detection.

8.1 Sentiment and Emotion Detection in SE

Sentiment and emotion have gained growing attention from the SE community, since they are demonstrated to be correlated with work performance and team collaboration [80]. Related text-based techniques, including sentiment and emotion detection techniques, have been widely applied in SE for enhancing software development, maintenance, and evolution [9, 32, 37, 40, 58, 69, 81, 82, 84, 95, 110, 111]. Many of these studies [40, 81, 82, 95] used the out-of-the-box sentiment or emotion detection tools (e.g., SentiStrength [100], NLTK [7], and Stanford NLP [68]) trained on non-technical texts. Among these tools, SentiStrength is considered to be the most widely adopted one in SE studies [17, 33, 39–41, 53, 77, 81, 93, 101]. However, some researchers noticed unreliable results when directly employing such tools for SE tasks [54, 61]. Jongeling et al. [54] observed the disagreement among these existing tools on the datasets in SE and found that the results of several SE studies involving these tools cannot be confirmed when a different tool is used. The negative results have inspired a series studies of SE-customized sentiment and emotion detection techniques.

8.1.1 SE-customized Sentiment Detection. Islam and Zibran [48] identified the challenges of SentiStrength in detecting sentiment in SE texts and developed a lexicon-based tool SentiStrength-SE to address the majority of the identified challenges. In addition, many feature-based machine learning methods were proposed, including SentiCR [1] and Senti4SD [10]. Different from these methods, SEntiMoji employs advanced word embedding and deep learning techniques. These techniques have also been used in previous relevant work. Biswas et al. [8] used word embeddings and deep learning for sentiment detection in SE. Different from this work, we employ word embedding and deep learning in a transfer learning way, which takes advantages of the prior efforts in other tasks. Similarly, Robbes and Janes [88] leveraged a pre-trained next word prediction model to tackle the scarcity of data in SE-customized sentiment detection. However, different from this work, our approach is developed based on a pre-trained emoji prediction model. Considering that expressing sentiment is the main function of emojis [45], the emoji prediction model can incorporate more sentiment knowledge than the next word prediction model, which can further facilitate the sentiment detection task. Recently, Lin et al. [60] proposed a pattern-based aspect sentiment detection method POME. The difference between POME and SEntiMoji mainly lies in two aspects. On one hand, POME aims to classify Stack Overflow sentences referring to APIs according to different aspects (e.g., performance and usability), and to determine their sentiment polarity. However, SEntiMoji and other aforementioned techniques do not involve aspect identification. However, POME is developed based on the patterns inferred by manually analyzing sentences from Stack Overflow linked to APIs. Therefore, POME is a specific method for analyzing opinions about APIs from Stack Overflow posts. Compared to POME, SEntiMoji is easier to be applied to other tasks in SE.

8.1.2 SE-customized Emotion Detection. To detect the emotional states of developers in the bi-directional emotional model, Islam and Zibran [50] proposed a dictionary-based tool DEVA, which can detect excitation, stress, depression, and relaxation from SE texts. Since the performance of DEVA is inherently limited by the quality and size of the dictionaries, Islam et al. [47] then proposed a machine learning method MarValous to address this limitation. The features used by MarValous are specifically defined for the four emotions that DEVA focuses on, so MarValous can also identify only those emotions. In addition, Calefato et al. [12] developed a feature-based machine learning method EmoTxt. Murgia et al. [74] constructed a machine learning classifier (namely ESEM-E in this article) to identify emotions in issue comments. The two methods as well as SEntiMoji can be used to detect more emotions than DEVA and MarValous. Moreover, different from EmoTxt and ESEM-E, SEntiMoji is developed based on word embedding, deep learning, and transfer learning techniques.

8.1.3 Benchmark Studies on Sentiment and Emotion Detection in SE. To assess the performance and reliability of the sentiment and emotion detection methods specifically customized for SE, researchers have performed several benchmark studies, where each method tackles the same datasets for comparison. For instance, for emotion detection, Islam et al. [47] compared MarValous and DEVA on JIRA and Stack Overflow comments. For sentiment detection, Ikram et al. [4] carried out an empirical study of applying three SE-customized sentiment detection tools (i.e., Sentistrength-SE, SentiCR, and Senti4SD) on code reviews. By comparison, Islam and Zibran [49] applied the SE-customized methods to three datasets, not a single dataset. They investigated that how well these methods really work on different datasets and which tool to choose in which context. Similarly, Novielli et al. [79] applied SE-customized sentiment detection methods on four benchmark datasets and investigated the impact of labeling approach on their performance. In line with these studies, we evaluate the performance of SEntiMoji through a benchmark study.

8.1.4 Difficulties in Sentiment Detection in SE. In previous studies, Islam and Zibran [48] and Novielli et al. [79] have identified difficulties faced by sentiment detection in SE through qualitative analysis. Specifically, Islam and Zibran [48] analyzed the samples misclassified by SentiStrength, while Novielli et al. [79] analyzed the samples for which SentiStrength-SE, Senti4SD, and SentiCR all yielded a wrong classification. By comparison, we identify difficulties (i.e., error causes) by analyzing the samples misclassified by SEntiMoji. Since the three studies distill difficulties based on different samples and the identification process may be affected by the subjectivity of coders, the findings may have differences. To elaborate on the comparison with existing literature, we analyze the similarity and dissimilarity between the findings of us and the previous two studies.

First, we compare our findings with the difficulties identified by Islam and Zibran [48]. Since Islam and Zibran [48] derived difficulties based on the samples misclassified by lexicon-based SentiStrength, most difficulties identified by them are summarized in the word level, i.e., what difficulties sentiment detection has in dealing with specific kinds of words. For instance, domain-specific meanings of words, context-sensitive variations in meanings of words, and misinterpretation of the letter 'X' are the most frequent difficulties they observed. By comparison, we mainly summarized the difficulties in dealing with specific kinds of sentences/documents, such as those with misinterpreted purposes (i.e., C2) and those containing complex context information (i.e., C3). However, there are also similarities between the findings of the two studies. Islam and Zibran [48] found that it is difficult to dealing with irony and sarcasm, which can be included in *C7: Figurative language*. Besides irony and sarcasm, C7 also includes the usage of other figurative languages such as metaphor. In addition, Islam and Zibran [48] found that it is difficult to detect subtle expression of sentiments, which is also observed in our results (i.e., *C1: Implicit sentiment*).

Then we compare our findings with the error causes identified by Novielli et al. [79]. To ease the comprehension, we list the error causes identified by them with the exact naming from the corresponding publication [79]:

(1) *Polar facts but neutral sentiment.* This error cause is included in our identified *C2: Misinterpreted purposes*. Besides the polar facts, C2 also includes neutral requests, questions, and tips that are mistaken as positive and negative.
(2) *General Error.* This error cause is related to the tool inability to deal with some textual cues or errors in pre-processing raw text and thus can map to *C6: Data pre-processing*.
(3) *Politeness.* This error cause is also observed in our results (i.e., *C5: Politeness*).
(4) *Implicit sentiment polarity.* This cause can be directly linked to *C1: Implicit sentiment*.
(5) *Subjectivity in sentiment annotation.* This error cause is not observed in our results. It can be attributed to the subjectivity of coders, which has been discussed in the threats to validity.
(6) *Inability of classifiers to deal with pragmatics or context information.* This error cause maps to *C3: Complex context information*.
(7) *Figurative language.* This error cause corresponds to *C7: Figurative language*.

From the comparison, we can find a substantial overlapping between the error causes identified by Novielli et al. [79] and us, which partly demonstrates the validity of our findings. In addition, we also report some unique error causes that have not been observed by Novielli et al. [79], such as neutral requests, questions, and tips included in C1. We also report that although SEntiMoji can outperform the existing methods, it still fails in capturing obvious sentiment in some cases (i.e, *C4: Failure to capture obvious sentiment*), which is not included in the findings of Novielli et al. [79]. Furthermore, Novielli et al. [79] only analyzed the error causes of sentiment detection in SE, but we also identified the error causes of emotion detection, where a unique error cause (i.e., *C8:*

The same emotion polarity) is observed. Overall, we believe that our work advances the knowledge over the previous work by Novielli et al. [79].

8.2 Emojis in Sentiment and Emotion Detection

Traditional sentiment and emotion detection in NLP is mainly performed in *unsupervised* or *supervised* ways. Unsupervised tools (e.g., SentiStrength) simply make use of lists of words annotated with sentiment polarity to determine the overall sentiment/emotion of a given text. However, fixed word lists cannot cope with the dynamic nature of the natural language [34]. Then researchers started to use labeled texts to train sentiment/emotion classifiers in a supervised way, where deep learning techniques are widely adopted [26, 116]. However, it is time-consuming to manually annotate texts on a large scale, thus resulting in a scarcity of labeled data. To tackle this problem, many researchers attempted to perform sentiment or emotion detection in a *distantly supervised* way. For example, they used emoticons [63] and specific hashtags [22] as a proxy for the emotional contents of texts. Recent studies [16, 30] extended the distant supervision to emojis, a more diverse set of indications, and demonstrated the superiority of emojis compared to emoticons. As emojis are becoming increasingly popular [2, 15, 65] and have the ability to express emotions [45], they are considered benign indications of sentiments and emotions [16, 30]. The emotional information contained in the emoji usage data can supplement the limited manually labeled data. In this work, we also employ emojis as indications of sentiment/emotion and tackle sentiment and emotion detection tasks using deep learning in a distantly supervised way. However, different from previous work [30], we apply such a method on SE-related texts and propose an SE-customized approach.

Recently, to address the challenge of sentiment and emotion detection in SE, researchers also started to analyze emoticons and emojis in software development platforms so as to find some potential solutions. Claes et al. [18] investigated the use of emoticons in open source software development. Lu et al. [66] analyzed the emoji usage on GitHub and found that emojis are often used to express sentiment on this platform. Furthermore, Imtiaz et al. [46] directly used emojis as the indicators of developers' sentiment on GitHub. Calefato et al. [10] and Ding et al. [25] took emoticons into account in their proposed SE-customized sentiment detection techniques. All of them demonstrated the feasibility of leveraging these emotional cues to benefit sentiment and emotion detection in SE. Following this line of research, this study leverages large-scale emoji usage from both technical and open domains to address sentiment and emotion detection in SE.

9 CONCLUSION

In this study, we have proposed SEntiMoji, an emoji-powered transfer learning approach for sentiment and emotion detection in SE. It is developed based on an existing representation model called DeepMoji. DeepMoji is pre-trained on Tweets and can represent texts with sentiment- and emotion-aware vectors. As technical knowledge is highlighted in current SE-customized sentiment and emotion detection, we also use GitHub data to incorporate more technical knowledge into DeepMoji. Then the fine-tuned representation model, as well as manually labeled data, is used to train the final sentiment/emotion classifier.

We evaluate the effectiveness of SEntiMoji on five benchmark datasets covering 10,096 samples for sentiment detection and four benchmark datasets covering 10,595 samples for emotion detection. In addition, for both sentiment and emotion detection, we use four representative methods as baselines to compare their performance against SEntiMoji. Results show that SEntiMoji can outperform existing sentiment and emotion detection methods with an average increase of 0.036 and 0.049 in macro-F1, respectively. Furthermore, we investigate the impact of Tweets, GitHub data, and labeled data on the performance of SEntiMoji and demonstrate the importance of the large-scale Tweets. Finally, we perform a qualitative analysis to distill the errors causes of

SEntiMoji, which suggests immediate actions and future research directions. To facilitate replications or other types of future work, we have released the data, scripts, trained models, and experimental results used in this study on https://github.com/SEntiMoji/SEntiMoji.

In the future, we plan to further improve SEntiMoji based on the identified error causes and limitations. Using SEntiMoji and its future releases, we also plan to analyze the quality of different software artifacts (e.g., APIs and libraries) by mining opinions posted by developers while discussing on Q&A websites such as Stack Overflow. Then we can leverage such crowd-sourced knowledge to build high-quality tools to recommend software artifacts to developers. In addition, we plan to verify the relationship between developers' emotional states and their outcome (e.g., productivity and bugs) by operating SEntiMoji on industrial datasets.

ACKNOWLEDGMENTS

The authors would like to thank Prof. Gang Huang from Peking University for his advice on detecting the impact of sentiment and emotion for development tasks in software engineering.

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'17)*. 106–111.

[2] Wei Ai, Xuan Lu, Xuanzhe Liu, Ning Wang, Gang Huang, and Qiaozhu Mei. 2017. Untangling emoji popularity through semantic embeddings. In *Proceedings of the 11th International Conference on Web and Social Media (ICWSM'17)*. 2–11.

[3] Akiko Aizawa. 2003. An information-theoretic perspective of TF-IDF measures. *Inf. Process. Manage.* 39, 1 (2003), 45–65.

[4] Ikram El Asri, Noureddine Kerzazi, Gias Uddin, Foutse Khomh, and Mohammed Amine Janati Idrissi. 2019. An empirical study of sentiments in code reviews. *Inf. Softw. Technol.* 114 (2019), 37–54.

[5] Lisa Feldman Barrett. 1998. Discrete emotions or dimensions? The role of valence focus and arousal focus. *Cogn. Emot.* 12, 4 (1998), 579–599.

[6] Yoav Benjamini and Daniel Yekutieli. 2001. The control of the false discovery rate in multiple testing under dependency. *Ann. Stat.* 29, 4 (2001), 1165–1188.

[7] Steven Bird and Edward Loper. 2004. NLTK: the natural language toolkit. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, Barcelona (ACL'04)*. 31.

[8] Eeshita Biswas, K. Vijay-Shanker, and Lori L. Pollock. 2019. Exploring word embedding techniques to improve sentiment analysis of software engineering texts. In *Proceedings of the 16th International Conference on Mining Software Repositories (MSR'19)*. 68–78.

[9] Cássio Castaldi Araujo Blaz and Karin Becker. 2016. Sentiment analysis in tickets for IT support. In *Proceedings of the 13th International Conference on Mining Software Repositories (MSR'16)*. 235–246.

[10] Fabio Calefato, Filippo Lanubile, Federico Maiorano, and Nicole Novielli. 2018. Sentiment polarity detection for software development. *Emp. Softw. Eng.* 23, 3 (2018), 1352–1382.

[11] Fabio Calefato, Filippo Lanubile, and Nicole Novielli. 2017. EmoTxt: A toolkit for emotion recognition from text. In *Proceedings of the 7th International Conference on Affective Computing and Intelligent Interaction Workshops and Demos (ACII Workshops'17)*. 79–80.

[12] Fabio Calefato, Filippo Lanubile, Nicole Novielli, and Luigi Quaranta. 2019. EMTk: the emotion mining toolkit. In *Proceedings of the 4th International Workshop on Emotion Awareness in Software Engineering (SEmotion@ICSE'19)*. 34–37.

[13] José Cambronero, Hongyu Li, Seohyun Kim, Koushik Sen, and Satish Chandra. 2019. When deep learning met code search. In *Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/SIGSOFT FSE'19)*. 964–974.

[14] Zhenpeng Chen, Yanbin Cao, Xuan Lu, Qiaozhu Mei, and Xuanzhe Liu. 2019. SEntiMoji: An emoji-powered learning approach for sentiment analysis in software engineering. In *Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/SIGSOFT FSE'19)*. 841–852.

[15] Zhenpeng Chen, Xuan Lu, Wei Ai, Huoran Li, Qiaozhu Mei, and Xuanzhe Liu. 2018. Through a gender lens: Learning usage patterns of emojis from large-scale Android users. In *Proceedings of the 2018 World Wide Web Conference (WWW'18)*. 763–772.

[16] Zhenpeng Chen, Sheng Shen, Ziniu Hu, Xuan Lu, Qiaozhu Mei, and Xuanzhe Liu. 2019. Emoji-powered representation learning for cross-lingual sentiment classification. In *Proceedings of the 2019 World Wide Web Conference on World Wide Web (WWW'19)*. 251–262.

[17] Shaiful Alam Chowdhury and Abram Hindle. 2016. Characterizing energy-aware software projects: are they different? In *Proceedings of the 13th International Conference on Mining Software Repositories (MSR'16)*. 508–511.

[18] Maëlick Claes, Mika Mäntylä, and Umar Farooq. 2018. On the use of emoticons in open source software development. In *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM'18)*. 50:1–50:4.

[19] Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educ. Psychol. Meas.* 20, 1 (1960), 37–46.

[20] Cristian Danescu-Niculescu-Mizil, Moritz Sudhof, Dan Jurafsky, Jure Leskovec, and Christopher Potts. 2013. A computational approach to politeness with application to social factors. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL'13)*. 250–259.

[21] Jason W. Davey, P. Cristian Gugiu, and Chris L. S. Coryn. 2010. Quantitative methods for estimating the reliability of qualitative data. *J. MultiDiscipl. Eval.* 6, 13 (2010), 140–162.

[22] Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Enhanced sentiment learning using Twitter hashtags and smileys. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING'10)*. 241–249.

[23] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT'19)*. 4171–4186.

[24] Thomas G. Dietterich. 1998. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Comput.* 10, 7 (1998), 1895–1923.

[25] 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'18)*. 7–13.

[26] Hai Ha Do, P. W. C. Prasad, Angelika Maag, and Abeer Alsadoon. 2019. Deep learning for aspect-based sentiment analysis: A comparative review. *Expert Syst. Appl.* 118 (2019), 272–299.

[27] Zi-Yi Dou. 2017. Capturing user and product information for document level sentiment analysis with deep memory network. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP'17)*. 521–526.

[28] Vasiliki Efstathiou, Christos Chatzilenas, and Diomidis Spinellis. 2018. Word embeddings for the software engineering domain. In *Proceedings of the 15th International Conference on Mining Software Repositories (MSR'18)*. 38–41.

[29] Manaal Faruqui, Yulia Tsvetkov, Dani Yogatama, Chris Dyer, and Noah A. Smith. 2015. Sparse overcomplete word vector representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL'15)*. 1491–1500.

[30] Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP'17)*. 1615–1625.

[31] Jerome H. Friedman. 2002. Stochastic gradient boosting. *Computational Statistics & Data Analysis* 38, 4 (2002), 367–378.

[32] Daviti Gachechiladze, Filippo Lanubile, Nicole Novielli, and Alexander Serebrenik. 2017. Anger and its direction in collaborative software development. In *Proceedings of the 39th IEEE/ACM International Conference on Software Engineering: New Ideas and Emerging Technologies Results Track (ICSE-NIER'17)*. 11–14.

[33] David García, Marcelo Serrano Zanetti, and Frank Schweitzer. 2013. The role of emotions in contributors activity: A case study on the GENTOO community. In *Proceedings of the 2013 International Conference on Cloud and Green Computing (CGC'13)*. 410–417.

[34] Anastasia Giachanou and Fabio Crestani. 2016. Like it or not: A survey of Twitter sentiment analysis methods. *Comput. Surv.* 49, 2 (2016), 28:1–28:41.

[35] Daniela Girardi, Filippo Lanubile, Nicole Novielli, Luigi Quaranta, and Alexander Serebrenik. 2019. Towards recognizing the emotions of developers using biometrics: the design of a field study. In *Proceedings of the 4th International Workshop on Emotion Awareness in Software Engineering*. 13–16.

[36] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.

[37] Daniel Graziotin, Fabian Fagerholm, Xiaofeng Wang, and Pekka Abrahamsson. 2018. What happens when software developers are (un)happy. *J. Syst. Softw.* 140 (2018), 32–47.

[38] Jin Guo, Jinghui Cheng, and Jane Cleland-Huang. 2017. Semantically enhanced software traceability using deep learning techniques. In *Proceedings of the 39th International Conference on Software Engineering (ICSE'17)*. 3–14.
[39] Emitza Guzman, David Azócar, and Yang Li. 2014. Sentiment analysis of commit comments in GitHub: an empirical study. In *Proceedings of the 11th Working Conference on Mining Software Repositories (MSR'14)*. 352–355.
[40] Emitza Guzman and Bernd Bruegge. 2013. Towards emotional awareness in software development teams. In *Proceedings of the Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE'13)*. 671–674.
[41] Emitza Guzman and Walid Maalej. 2014. How do users like this feature? A fine grained sentiment analysis of app reviews. In *Proceedings of the IEEE 22nd International Requirements Engineering Conference (RE'14)*. 153–162.
[42] Maryam Hasan, Elke Rundensteiner, and Emmanuel Agu. 2014. EMOTEX: Detecting emotions in twitter messages. In *Proceedings of the ASE BIGDATA/SOCIALCOM/CYBERSECURITY Conference (ASE'14)*. 27–31.
[43] M. Hermans and B. Schrauwen. 2013. Training and analysing deep recurrent neural networks. In *Proceedings of the 27th Annual Conference on Neural Information Processing (NIPS'13)*. 190–198.
[44] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780.
[45] Tianran Hu, Han Guo, Hao Sun, Thuy-vy Thi Nguyen, and Jiebo Luo. 2017. Spice up your chat: the intentions and sentiment effects of using emojis. In *Proceedings of the 11th International Conference on Web and Social Media (ICWSM'17)*. 102–111.
[46] Nasif Imtiaz, Justin Middleton, Joymallya Chakraborty, Neill Robson, Gina Bai, and Emerson R. Murphy-Hill. 2019. Investigating the effects of gender bias on GitHub. In *Proceedings of the 41st International Conference on Software Engineering (ICSE'19)*. 700–711.
[47] Md Rakibul Islam, Md Kauser Ahmmed, and Minhaz F. Zibran. 2019. MarValous: Machine learning based detection of emotions in the valence-arousal space in software engineering text. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing (SAC'19)*. 1786–1793.
[48] Md Rakibul Islam and Minhaz F. Zibran. 2017. Leveraging automated sentiment analysis in software engineering. In *Proceedings of the 14th International Conference on Mining Software Repositories (MSR'17)*. 203–214.
[49] Md Rakibul Islam and Minhaz F. Zibran. 2018. A comparison of software engineering domain specific sentiment analysis tools. In *Proceedings of the 25th International Conference on Software Analysis, Evolution and Reengineering (SANER'18)*. 487–491.
[50] Md Rakibul Islam and Minhaz F. Zibran. 2018. DEVA: sensing emotions in the valence arousal space in software engineering text. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing (SAC'18)*. 1536–1543.
[51] Md Rakibul Islam and Minhaz F. Zibran. 2018. SentiStrength-SE: Exploiting domain specificity for improved sentiment analysis in software engineering text. *J. Syst. Softw.* 145 (2018), 125–146.
[52] Pooyan Jamshidi, Norbert Siegmund, Miguel Velez, Christian Kästner, Akshay Patel, and Yuvraj Agarwal. 2017. Transfer learning for performance modeling of configurable systems: an exploratory analysis. In *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE'17)*. 497–508.
[53] Robbert Jongeling, Subhajit Datta, and Alexander Serebrenik. 2015. Choosing your weapons: On sentiment analysis tools for software engineering research. In *Proceedings of the 2015 IEEE International Conference on Software Maintenance and Evolution (ICSME'15)*. 531–535.
[54] Robbert Jongeling, Proshanta Sarkar, Subhajit Datta, and Alexander Serebrenik. 2017. On negative results when using sentiment analysis tools for software engineering research. *Emp. Softw. Eng.* 22, 5 (2017), 2543–2584.
[55] Francisco Jurado and Pilar Rodríguez Marín. 2015. Sentiment analysis in monitoring software development processes: an exploratory case study on GitHub's project issues. *J. Syst. Softw.* 104 (2015), 82–89.
[56] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR'15)*.
[57] Ekrem Kocaguneli, Tim Menzies, and Emilia Mendes. 2015. Transfer learning in effort estimation. *Emp. Softw. Eng.* 20, 3 (2015), 813–843.
[58] Miikka Kuutila, Mika V. Mäntylä, Maëlick Claes, Marko Elovainio, and Bram Adams. 2018. Using experience sampling to link software repositories with emotions and work well-being. In *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM'18)*. 29:1–29:10.
[59] Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Trans. Assoc. Comput. Ling.* 3 (2015), 211–225.
[60] Bin Lin, Fiorella Zampetti, Gabriele Bavota, Massimiliano Di Penta, and Michele Lanza. 2019. Pattern-based mining of opinions in Q&A websites. In *Proceedings of the 41st International Conference on Software Engineering (ICSE'19)*. 548–559.
[61] 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 (ICSE'18)*. 94–104.

[62] Bing Liu. 2012. *Sentiment Analysis and Opinion Mining*. Morgan & Claypool Publishers.
[63] Kun-Lin Liu, Wu-Jun Li, and Minyi Guo. 2012. Emoticon smoothed language models for twitter sentiment analysis. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI'12)*.
[64] Nikola Ljubesic and Darja Fiser. 2016. A global analysis of emoji usage. In *Proceedings of the 10th Web as Corpus Workshop (WAC@ACL'16)*. 82–89.
[65] Xuan Lu, Wei Ai, Xuanzhe Liu, Qian Li, Ning Wang, Gang Huang, and Qiaozhu Mei. 2016. Learning from the ubiquitous language: An empirical analysis of emoji usage of smartphone users. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp'16)*. 770–780.
[66] Xuan Lu, Yanbin Cao, Zhenpeng Chen, and Xuanzhe Liu. 2018. A first look at emoji usage on GitHub: An empirical study. arxiv:1812.04863. Retrieved from http://arxiv.org/abs/1812.04863.
[67] Howard Lune and Bruce L. Berg. 2016. *Qualitative Research Methods for the Social Sciences*. Pearson Higher Ed.
[68] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford coreNLP natural language processing toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL'14)*. 55–60.
[69] 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'16)*. 247–258.
[70] Mika V. Mäntylä, Nicole Novielli, Filippo Lanubile, Maëlick Claes, and Miikka Kuutila. 2017. Bootstrapping a lexicon for emotional arousal in software engineering. In *Proceedings of the 14th International Conference on Mining Software Repositories (MSR'17)*. 198–202.
[71] Ali Mesbah, Andrew Rice, Emily Johnston, Nick Glorioso, and Edward Aftandilian. 2019. DeepDelta: Learning to repair compilation errors. In *Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/SIGSOFT FSE'19)*. 925–936.
[72] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of the 1st International Conference on Learning Representations (ICLR'13)*.
[73] Myriam Munezero, Calkin Suero Montero, Erkki Sutinen, and John Pajunen. 2014. Are they different? Affect, feeling, emotion, sentiment, and opinion detection in text. *IEEE Trans. Affect. Comput.* 5, 2 (2014), 101–111.
[74] Alessandro Murgia, Marco Ortu, Parastou Tourani, Bram Adams, and Serge Demeyer. 2018. An exploratory qualitative and quantitative analysis of emotions in issue report comments of open source systems. *Emp. Softw. Eng.* 23, 1 (2018), 521–564.
[75] Alessandro Murgia, Parastou Tourani, Bram Adams, and Marco Ortu. 2014. Do developers feel emotions? An exploratory analysis of emotions in software artifacts. In *Proceedings of the 11th Working Conference on Mining Software Repositories (MSR'14*). 262–271.
[76] Jaechang Nam, Sinno Jialin Pan, and Sunghun Kim. 2013. Transfer defect learning. In *Proceedings of the 35th International Conference on Software Engineering (ICSE'13)*. 382–391.
[77] 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'15)*. 33–40.
[78] Nicole Novielli, Fabio Calefato, and Filippo Lanubile. 2018. A gold standard for emotion annotation in stack overflow. In *Proceedings of the 15th International Conference on Mining Software Repositories (MSR'18)*. 14–17.
[79] Nicole Novielli, Daniela Girardi, and Filippo Lanubile. 2018. A benchmark study on sentiment analysis for software engineering research. In *Proceedings of the 15th International Conference on Mining Software Repositories (MSR'18)*. 364–375.
[80] Nicole Novielli and Alexander Serebrenik. 2019. Sentiment and emotion in software engineering. *IEEE Softw.* 36, 5 (2019), 6–9.
[81] 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 *Proceedings of the 12th IEEE/ACM Working Conference on Mining Software Repositories (MSR'15)*. 303–313.
[82] Marco Ortu, Giuseppe Destefanis, Steve Counsell, Stephen Swift, Roberto Tonelli, and Michele Marchesi. 2016. Arsonists or firefighters? Affectiveness in agile software development. In *Proceedings of the 2016 International Conference on Agile Software Development (XP'16)*. Springer, 144–155.
[83] Marco Ortu, Alessandro Murgia, Giuseppe Destefanis, Parastou Tourani, Roberto Tonelli, Michele Marchesi, and Bram Adams. 2016. The emotional side of software developers in JIRA. In *Proceedings of the 13th International Conference on Mining Software Repositories (MSR'16*). 480–483.
[84] Sebastiano Panichella, Andrea Di Sorbo, Emitza Guzman, Corrado Aaron Visaggio, Gerardo Canfora, and Harald C. Gall. 2015. How can i improve my app? Classifying user reviews for software maintenance and evolution. In *Proceedings of the 2015 IEEE International Conference on Software Maintenance and Evolution (ICSME'15*). 281–290.

[85] 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'14)*. 1532–1543.

[86] 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'18)*. 2227–2237.

[87] Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. 2011. Classifier chains for multi-label classification. *Mach. Learn.* 85, 3 (2011), 333–359.

[88] Romain Robbes and Andrea Janes. 2019. Leveraging small software engineering data sets with pre-trained neural networks. In *Proceedings of the 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE (NIER)'19)*. 29–32.

[89] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1988. Learning representations by back-propagating errors. *Nature* 323, 6088 (1988), 533–536.

[90] James A. Russell and Albert Mehrabian. 1977. Evidence for a three-factor theory of emotions. *J. Res. Pers.* 11, 3 (1977), 273–294.

[91] Fabrizio Sebastiani. 2002. Machine learning in automated text categorization. *Comput. Surv.* 34, 1 (2002), 1–47.

[92] Phillip Shaver, Judith Schwartz, Donald Kirson, and Cary O'connor. 1987. Emotion knowledge: Further exploration of a prototype approach.*J. Pers. Soc. Psychol.* 52, 6 (1987), 1061.

[93] 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'16)*. 520–523.

[94] Tom De Smedt and Walter Daelemans. 2012. Pattern for python. *J. Mach. Learn. Res.* 13, 1 (2012), 2063–2067.

[95] Rodrigo Souza and Bruno Silva. 2017. Sentiment analysis of travis CI builds. In *Proceedings of the 14th International Conference on Mining Software Repositories (MSR'17)*. 459–462.

[96] Newton Spolaôr, Everton Alvares Cherman, Maria Carolina Monard, and Huei Diana Lee. 2012. A comparison of multi-label feature selection methods using the problem transformation approach. In *Proceedings of the XXXVIII Latin American Computer Conference—Selected Papers (CLEI'12 Selected Papers)*. 135–151.

[97] Carlo Strapparava and Alessandro Valitutti. 2004. WordNet affect: An affective extension of wordnet. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC'04)*.

[98] Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Conference of the Association for Computational Linguistics (ACL'19)*. 3645–3650.

[99] Johan A. K. Suykens and Joos Vandewalle. 1999. Least squares support vector machine classifiers. *Neural Process. Lett.* 9, 3 (1999), 293–300.

[100] Mike Thelwall, Kevan Buckley, Georgios Paltoglou, Di Cai, and Arvid Kappas. 2010. Sentiment in short strength detection informal text. *J. Assoc. Inf. Sci. Technol.* 61, 12 (2010), 2544–2558.

[101] Parastou Tourani and Bram Adams. 2016. The impact of human discussions on just-in-time quality assurance: An empirical study on OpenStack and Eclipse. In *Proceedings of the IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER'16)*. 189–200.

[102] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the Annual Conference on Neural Information Processing Systems 2017 (NIPS'17)*. 6000–6010.

[103] Bin Wang, Angela Wang, Fenxiao Chen, Yuncheng Wang, and C-C Jay Kuo. 2019. Evaluating word embedding models: Methods and experimental results. *APSIPA Trans. Sign. Inf. Process.* 8 (2019).

[104] Yi Wang. 2019. Emotions extracted from text vs. true emotions-an empirical evaluation in SE context. In *Proceedings of the 34th IEEE/ACM International Conference on Automated Software Engineering (ASE'19)*. 230–242.

[105] Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. 2016. Attention-based LSTM for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP'16)*. 606–615.

[106] Amy Beth Warriner, Victor Kuperman, and Marc Brysbaert. 2013. Norms of valence, arousal, and dominance for 13,915 english lemmas. *Behav. Res. Methods* 45, 4 (2013), 1191–1207.

[107] Huihui Wei and Ming Li. 2017. Supervised deep features for software functional clone detection by exploiting lexical and syntactical information in source code. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI'17)*. 3034–3040.

[108] Honghao Wei, Fuzheng Zhang, Nicholas Jing Yuan, Chuan Cao, Hao Fu, Xing Xie, Yong Rui, and Wei-Ying Ma. 2017. Beyond the words: Predicting user personality from heterogeneous information. In *Proceedings of the 10th ACM International Conference on Web Search and Data Mining (WSDM'17)*. 305–314.

[109] Theresa Wilson. 2008. Annotating subjective content in meetings. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC'08)*.

[110] Michal R. Wróbel. 2013. Emotions in the software development process. In *Proceedings of the 6th International Conference on Human System Interactions (HSI'13)*. 518–523.

[111] Michal R. Wrobel. 2016. Towards the participant observation of emotions in software development teams. In *Proceedings of the 2016 Federated Conference on Computer Science and Information Systems (FedCSIS'16)*. 1545–1548.

[112] Jiacheng Xu, Danlu Chen, Xipeng Qiu, and Xuanjing Huang. 2016. Cached long short-term memory neural networks for document-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP'16)*. 1660–1669.

[113] Min Yang, Wenting Tu, Jingxuan Wang, Fei Xu, and Xiaojun Chen. 2017. Attention based LSTM for target dependent sentiment classification. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI'17)*. 5013–5014.

[114] Xin Ye, Hui Shen, Xiao Ma, Razvan C. Bunescu, and Chang Liu. 2016. From word embeddings to document similarities for improved information retrieval in software engineering. In *Proceedings of the 38th International Conference on Software Engineering (ICSE'16)*. 404–415.

[115] Dani Yogatama, Manaal Faruqui, Chris Dyer, and Noah A. Smith. 2015. Learning word representations with hierarchical sparse coding. In *Proceedings of the 32nd International Conference on Machine Learning (ICML'15)*. 87–96.

[116] Lei Zhang, Shuai Wang, and Bing Liu. 2018. Deep learning for sentiment analysis: A survey. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* 8, 4 (2018).

[117] Xinjie Zhou, Xiaojun Wan, and Jianguo Xiao. 2016. Attention-based LSTM network for cross-lingual sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP'16)*. 247–256.

Received December 2019; revised July 2020; accepted September 2020