Can We Use SE-specific Sentiment Analysis Tools in a Cross-Platform Setting?

Nicole Novielli, Fabio Calefato, Davide Dongiovanni, Daniela Girardi, Filippo Lanubile
University of Bari, Italy
nicole.novielli,fabio.calefato,daniela.girardi,filippo.lanubile@uniba.it
d.dongiovanni@studenti.uniba.it

ABSTRACT

In this paper, we address the problem of using sentiment analysis tools 'off-the-shelf', that is when a gold standard is not available for retraining. We evaluate the performance of four SE-specific tools in a cross-platform setting, i.e., on a test set collected from data sources different from the one used for training. We find that (i) the lexicon-based tools outperform the supervised approaches retrained in a cross-platform setting and (ii) retraining can be beneficial in within-platform settings in the presence of robust gold standard datasets, even using a minimal training set. Based on our empirical findings, we derive guidelines for reliable use of sentiment analysis tools in software engineering.

CCS CONCEPTS

• **Software and its engineering**; • **Information systems** → **Sentiment analysis**; • **Computing methodologies** → *Machine learning*; • **Human-centered computing** → Collaborative and social computing;

KEYWORDS

Sentiment analysis, empirical software engineering, human factors, NLP, machine learning

ACM Reference Format:
Nicole Novielli, Fabio Calefato, Davide Dongiovanni, Daniela Girardi, Filippo Lanubile. 2020. Can We Use SE-specific Sentiment Analysis Tools in a Cross-Platform Setting?. In *17th International Conference on Mining Software Repositories (MSR '20), October 5–6, 2020, Seoul, Republic of Korea.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3379597.3387446

1 INTRODUCTION

Investigating the role of affect has emerged as a consolidated trend of research on human aspects in software engineering [28, 31]. Sentiment analysis is used to detect emotions in social coding platforms, such as GitHub [15, 36, 43], issue-tracking tools, such as Jira [13, 25, 32], and collaborative knowledge-sharing sites, such as Stack Overflow [7]. Further research has been leveraging sentiment analysis for requirements elicitation based on opinion detection in user-generated content [14, 24].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MSR '20, October 5–6, 2020, Seoul, Republic of Korea
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-7517-7/20/05…$15.00
https://doi.org/10.1145/3379597.3387446

Despite the popularity of general-purpose sentiment analysis tools, a consensus has been reached in the research community about the negative results obtained when using such tools 'off-the-shelf' to detect developers' emotions [19, 23, 29], thus indicating the need for fine-tuning such tools for the software engineering domain [26]. Trying to overcome these limitations, researchers have started to build their own classifiers specifically customized for software engineering (SE) [5, 11, 13, 17, 18]. Among others, four SE-specific tools are publicly available, namely Senti4SD [6], SentiStrength-SE [17], SentiCR [2], and DEVA [18]. SentiStrength-SE and DEVA implement a lexicon-based approach and have been optimized on a gold standard dataset of manually annotated content from Jira. Conversely, Senti4SD and SentiCR implement a supervised approach for training polarity classification models. They also offer retraining functions for model optimization and fine-tuning based on a custom gold standard dataset.

In a previous benchmarking study [30], we compared the predictions of Senti4SD, SentiCR, and SentiStrength-SE, showing how domain-specific customization provides a boost in accuracy compared to the baseline approach represented by SentiStrength [47], an off-the-shelf tool trained and validated on general-purpose social media. Specifically, the best performance was observed for tools implementing supervised approaches. Based on this evidence, customized retraining of the classifiers was recommended. We executed the study in a within-platform setting, that is, we trained and tested each classifier using a gold standard from the same data source. However, building a manually annotated gold standard is a time-consuming task and, as such, not always feasible.

In this paper, we address the problem of using SE-specific sentiment analysis tools in a cross-platform setting, i.e., in the absence of a gold standard for a target data source. Our study builds upon the design and results of two previous studies, one by Jongeling et al. [19] and one of our previous works [30], assessing the performance of general-purpose and SE-specific sentiment analysis tools, respectively. Specifically, in line with these previous studies [19, 30], we define the following research questions:

- *RQ1* - To what extent do different SE-specific sentiment analysis tools agree with the emotions of software developers when used as 'off-the-shelf' tools in a cross-platform setting?
- *RQ2* - To what extent do results from different SE-specific sentiment analysis tools agree with each other when used as 'off-the-shelf' tools in a cross-platform setting?

To enable the comparison with previous research, we assess the tool performance on two gold standard datasets from the software engineering domain, namely a Jira dataset of 6K comments [33] and a Stack Overflow dataset of 4K posts (questions, answers, and comments). Both datasets have been manually annotated by adopting a

model-driven approach, that is by referring to theoretical emotion models translated into detailed guidelines for the human raters. Specifically, the annotation of the Jira and Stack Overflow dataset is based on the theoretical model of emotions defined by Shaver et al. [42]. Furthermore, we create and include in our benchmark a gold standard dataset of 7K comments from GitHub pull-request and commit comments. The GitHub comments have been manually annotated by three of the authors, following the same annotation guidelines adopted for the Stack Overflow and Jira datasets.

Finally, we aim to understand how many training documents are required so that choosing to retrain a supervised tool is convenient compared to using a lexicon-based approach 'off-the-shelf'. Indeed, building a manually annotated gold standard for sentiment analysis is a time-consuming task that requires careful training of the raters and the appropriate choice of the emotion model [29]. As such, we formulate a third research question:

- *RQ*3 - To what extent is the performance of SE-specific sentiment analysis tools affected by the size of the training set?

The contributions of this paper are as follows. First, we release a dataset of more than 7k comments from GitHub.[1] To the best of our knowledge, this is the first publicly available dataset including texts from GitHub annotated for sentiment polarity. As a second contribution, we enhance the current understanding of the most frequent causes of misclassification due to cross-platform use of sentiment analysis tools when applied in the software engineering domain. Finally, we derive empirically-based recommendations for the safe adoption of SE-specific tools for sentiment analysis, both in presence and in absence of a gold standard for retraining.

The paper is organized as follows. In Section 2, we address sentiment analysis in software engineering and summarize the previous benchmarking studies we build upon. In Section 3, we describe the SE-specific tools we include in our benchmark. In Section 4, we describe the three manually annotated gold standard dataset that we include in our benchmark, with a detailed description of the annotation study we conduct to build the GitHub gold standard. Then, we describe the study design in Section 5, report results in Section 6, and present guidelines for sentiment analysis in SE in Section 7. Finally, we discuss the threats to validity in Section 8 and conclude in Section 9.

2 SENTIMENT ANALYSIS IN SOFTWARE ENGINEERING

Sentiment analysis is the task of mining the positive or negative opinions and emotions conveyed by text [34]. Psychologists have worked for decades at the definition of theoretical model for emotions [21, 39, 42]. Regardless of the specific taxonomy, emotions can be mapped to the polarity dimension, i.e., classified as positive, negative, or neutral. This holds true also for other states of the affective spectrum, such as opinions, which are traditionally investigated by research in sentiment analysis.

Sentiment analysis is a consolidated research field, and a plethora of tools are nowadays publicly available for research purposes. In recent years, a trend emerged and consolidated to leverage sentiment analysis as a tool for empirical software engineering. Recent studies applied sentiment analysis to Stack Overflow, in order to define empirically-driven guidelines for successful question-writing in technical question and answering sites [7]. Users' sentiment in app reviews [24] or social media [14] was studied to support requirements elicitation. Developers' emotions were studied in the context of issue tracking to investigate their impact on issue-fixing time [27] or to understand how emotions are communicated in collaborative-software development [13]. Opinion mining on Stack Overflow was leveraged to support the development of recommender systems for software libraries [22, 48].

General-purpose sentiment analysis tools have been trained on movie reviews [44] or social media texts [47]. In spite of their popularity, there is a general consensus in the research community about the negative results obtained when using such tools in SE [19, 29]. In our previous work [29], we manually investigated a dataset of 800 posts from Stack Overflow, reporting domain-specific lexicon as the main cause for false positives in negative sentiment detection. Jongeling and colleagues [19] compared the predictions of widely used off-the-shelf sentiment analysis tools, showing not only how these tools disagree with human annotation of developers' emotions and opinions, but also how they disagree with each other. They conclude advocating in favor of SE-tuning of sentiment analysis tool to allow reliable empirical studies.

To overcome these limitations, researchers recently started to develop their own SE-specific tools [5, 11, 22, 35, 48]. At the time of this study, we identified four of the most widely used, SE-specific tools available for research use (see Section 3). By replicating the original study of Jongeling et al., our benchmark study [30] presented at MSR 2018 investigated to what extent fine-tuning sentiment analysis tools for the software engineering domain do succeed in improving the accuracy of emotion detection. The results show that fine-tuning of tools on SE-related text does improve the performance of sentiment classification, provided that the train set used for retraining are built following guidelines grounding on theoretical models of affect. The study was performed in a within-platform setting, that is the train and test sets used for assessing the performance of classifiers based on machine-learning were collected on the same collaborative development platform. In the current study, we partially replicate the experimental setting of the two previous studies [19, 30]. The goal is to further advance the state of the art by addressing the problem of using SE-specific tools in a cross-platform setting, that is when a gold standard dataset is not available for retraining.

3 SE-SPECIFIC SENTIMENT ANALYSIS TOOLS

State-of-the-art approaches to sentiment analysis treat subjectivity and polarity detection as text classification problems. The existing tools implement two main approaches. The first one exploits *machine learning* algorithms for training supervised classifiers based on textual features. Such features are typically based on words occurring in the documents (i.e., tokens, stems, lemmata) or syntactic features as part-of-speech tags. Often, textual features are extracted using n-grams, i.e., sequences of *n* contiguous words [37]. Such approaches mainly rely on state-of-the-art machine learning algorithms. Recently, researchers also leveraged deep learning [50]

[1] The dataset can be downloaded from: https://doi.org/10.6084/m9.figshare.11604597

in combination with emoji-based vector representations of documents [11]. On the other hand, *lexicon-based* methods [45] exploit the *prior* sentiment polarity of words in a text, that is the word positive, negative, or neutral polarity based on large lexicons of words annotated with scores indicating their positive or negative semantic orientation. The overall sentiment of a text is then computed based on the prior polarity of the words occurring in it. However, due to the effect of contextual valence shifters, such as intensifiers (e.g., adverbs as "very") or negations (e.g., "not"), the prior polarity of a given word might not match the actual sentiment of the author. Therefore, lexicon-based approaches are usually integrated with additional rules to adjust the prior polarity of words based on the effect of intensifiers and negations.

In this study, we assess the performance of four tools that are publicly available at the time of writing. Specifically, Senti4SD and SentiCR leverage supervised machine-learning, while SentiStrength-SE and DEVA implement a lexicon-based approach.

Senti4SD [6] is our own supervised polarity classifier, which leverages a suite of features based on Bag of Words (BoW), sentiment lexicons, and semantic features based on word embedding. Along with the toolkit, we distribute a classification model, trained and validated on a gold standard of about 4K questions, answers, and comments from Stack Overflow, and manually annotated for sentiment polarity. Furthermore, the toolkit provides a training method that enables the customization of the classifier using a gold standard as input. Compared to the performance obtained by SentiStrength on the same Stack Overflow test set, Senti4SD reduces the misclassifications of neutral and positive posts as emotionally negative (F1=.87). A good performance (F1=.84) is also achieved with a minimal set of training documents. For this study, we use the Python version of Senti4SD [8].

SentiCR [2] is a supervised tool that leverages a feature vector generated by computing term frequency–inverse document frequency (tf-idf) for Bag-of-Words (BoW) extracted from the input text. SentiCR implements basic preprocessing of the raw input text to expand contractions, handle negations and emoticons, remove stop-words, derive word stems, and remove code snippets. Furthermore, it performs SMOTE to handle the class imbalance in the training set. The currently distributed version implements a training approach based on Gradient Boosting Tree and requires a training set as an input in order to retrain the model and use it on the target document collection. A mean accuracy of 83%, a precision of .68, and a recall of .58 are reported on a gold standard of 2,000 code-review comments.

SentiStrength-SE [17] is built upon the API of SentiStrength [47]. It leverages a manually adjusted version of the SentiStrength lexicon and implements *ad hoc* heuristics to correct the misclassifications observed when running it on a subset of the dataset of Ortu et al. [33]. The sentiment scores of words in the lexicon were manually adjusted to reflect the semantics and neutral polarity of domain words such as "support" or "default." As a result, SentiStrength-SE outperforms SentiStrength on technical texts.

DEVA [18] leverages a lexicon-based approach for the identification of both emotion activation (arousal) and polarity from text. To this end, the tool uses two separate dictionaries developed by exploiting a general-purpose lexicon as well as one specific for software engineering text. To further increase its accuracy, DEVA also includes several heuristics, some of which are borrowed from SentiStrength-SE. For the empirical evaluation, a ground-truth dataset was built, consisting of 1,795 Jira issue comments, manually annotated by three human raters, on which DEVA was found to achieve a precision of .82 and a recall of .79.

Table 1: Datasets included in our benchmark, with distribution of polarity classes.

Dataset	Overall documents	Polarity Classes		
		Neutral	*Positive*	*Negative*
GitHub	7,122	3,022 (43%)	2,013 (28%)	2,087 (29%)
Jira [33]	5,869	3,955 (67%)	1,128 (19%)	786 (14%)
Stack Overflow [6]	4,423	1,694 (38%)	1,527 (35%)	1,202 (27%)

4 ANNOTATED DATASETS

The quality of the gold standard largely impacts the classification performance, regardless of the machine learning approach [1, 46]. As for sentiment analysis, we found that SE-specific customization might not guarantee a reasonable accuracy if *ad hoc* annotation is performed [30]. In fact, *ad hoc* annotation consists of asking the raters to provide polarity labels according to their subjective perception of the semantic orientation of the text [2, 23]. In our previous benchmarking study [30], we provide evidence that the absence of clear guidelines for annotation leads to noisy gold standards, thus resulting in unreliable model training and testing. As such, we argue that reliable sentiment analysis in software engineering is nonetheless possible, provided that manual annotation of gold standards is supported by theoretical models of emotion. In line with our previous findings, in this study, we employ two model-driven datasets from Stack Overflow and Jira (described next), consistently annotated according to the same theoretical framework [42]. As a third dataset, we manually labeled over 7K comments from GitHub pull requests and commits, following the same annotation schema and guidelines used in [6] as detailed next. In Table 1, we report the overall number of documents[2] included in each dataset with the distribution of labels for each polarity class.

The **Stack Overflow dataset** [6] consists of 4,423 posts, including questions, answers, and comments manually annotated with polarity labels by twelve trained coders with a background in Computer Science. The coders were trained to explicitly indicate a polarity label for each post according to the emotion detected, based on the labels included in the Shaver framework [42]. Each post was annotated by three raters and received the polarity gold label based on majority voting. The gold standard resulting from this procedure is well-balanced, with 35% of posts conveying positive emotions, 27% presenting negative emotions, and 38% of posts labeled as neutral, denoting the absence of emotions. A Cohen's [12] κ of .74 is observed, indicating a substantial inter-rater agreement [49].

The **Jira dataset** [33] includes about 6,000 issue comments and sentences authored by software developers of popular open-source software projects, such as Apache and Spring. The Jira dataset is

[2]In the remainder of the paper, we will use the term 'document' to refer to the text items (posts or comments) in our datasets.

originally distributed with the six emotion labels from the Shaver et al. framework [42] (i.e., love, joy, surprise, anger, fear, and sadness), whereas this study focuses on emotion polarity (i.e., the positive, negative, or neutral valence conveyed by texts). As such, we use an approach consistent with the labeling guidelines adopted for the Stack Overflow gold standard described above, thus resulting in two homogeneous benchmarking datasets grounded on the same emotion model. Specifically, we translate positive emotions, i.e., love and joy, into a positive polarity label. Similarly, sadness, anger, and fear are mapped to the negative polarity class. Instead, surprise cases are discarded as this emotion label could be either considered positive or negative, depending on the expectations of the author of a text. Finally, the absence of emotions defines neutral cases. Unlike the Stack Overflow dataset, the Jira gold standard is not well-balanced, with 19% of posts conveying positive emotions, 14% conveying negative emotions, and 67% labeled as neutral. The authors do not assess the κ agreement for the polarity classes, as they originally provide labels for discrete emotions. Conversely, they report the κ for the emotion annotation, with values ranging from absence of agreement for *anger* to moderate agreement for *love*, for which the highest value observed is $\kappa = .55$.

The **GitHub dataset** includes about 7,000 pull request and commit comments. The dataset is well-balanced, is a desirable property for a training set [16], with 28% and 29% of posts conveying positive and negative emotions, respectively. The remaining 43% of posts are labeled as neutral, as they do not convey emotions. The dataset has been annotated by three of the authors following the guidelines for annotation defined for the creation of the Stack Overflow dataset [6]. As a unit of analysis, we consider the entire comment, i.e., the raters were requested to annotate the sentiment conveyed by the whole comment. Specifically, the raters were trained to provide a polarity label based on the emotion detected according to the Shaver model, following the emotion-polarity mapping described for the Stack Overflow and Jira datasets.

We built our GitHub gold standard using the iterative approach depicted in Figure 1. Specifically, we designed the protocol for our annotation study following the methodology adopted in the study on anger in collaborative software development [13]. We extracted the annotation sample for each iteration from the dataset of comments created to study the sentiment of security discussion in GitHub [36]. We started with an annotation sample of 4k comments, randomly extracted from the initial dataset of 116k comments. Each comment was labeled by two raters independently. We observed an almost perfect inter-rater agreement ($\kappa = .84$). Once the individual annotation was completed, we assigned the manually provided gold label to all the comments for which the two raters agreed. Then, the three raters discussed the 340 disagreement cases in a plenary meeting: we include in the gold standard all those comments for which the initial disagreement is resolved through discussion (298) and discard the others (42, corresponding to 1.05% of the annotation sample). Furthermore, 27 duplicate comments were removed.

As a result of this first step, we obtained 3,931 comments for which the three raters agreed both on the presence of emotions and on its polarity. Given the unbalanced distribution of the obtained dataset (see Figure 1), we implemented the subsequent two annotation steps to collect more positive and negative comments. Since manual labeling is a time-consuming activity, we accelerated the process by leveraging a semi-automatic approach involving manual confirmation of automatically obtained polarity labels. Using the initial core of 3,931 comments, we retrained the polarity classification model using the Senti4SD toolkit, as it reported a better precision than SentiCR for both the positive and negative classes. Specifically, we observe a precision of .61 (Senti4SD) vs. .34 (SentiCR) for the negative class. Conversely, the precision for the positive class is comparable (.89 for Senti4SD vs. 88 for SentiCR). The reason behind this choice is that, by optimizing for precision, we reduce the number of neutral sentences misclassified as expressing sentiment, thus avoiding to annoy the raters with useless annotation of neutral cases. The performance of this classification model is reported in Figure 1 (Precision = .79, Recall =.59, F1-measure = .62).

In the second step, we applied this classifier to the remaining 112k comments of the original dataset by Pletea et al., excluding all cases that were already included in the first annotation sample. We obtained an automatically labeled dataset, from which we randomly extracted a new annotation sample of 600 positive and 600 negative comments. To avoid any bias, the annotators were not provided with the outcome of the classifier. As such, their annotation was done only based on the text, as in the first round. Again, the raters performed the annotation individually. They confirmed the classifier label for 343 positive and 550 negative comments. These new confirmed cases were added to the gold standard, resulting in an enriched set of 4,809 comments, of which 63% labeled as neutral, 19% as positive, and 18% as negative. To further enrich and balance the gold standard, we repeated the training with this new set, observing an improved performance of the classification model (Precision = .88, Recall =.82, F1-measure = .84). We use this second classifier to label the remaining 111k comments and repeat the manual confirmation step for 3,000 comments. This third annotation step resulted in 1,124 positive and 1,204 additional negative comments. The final GitHub gold standard includes 7,122 comments that we use for this study.

5 STUDY DESIGN

Experimental Setting. To answer RQ1 and RQ2, we assess the performance on the three gold standard datasets of the two supervised tools (Senti4SD and SentiCR), which can be retrained, and the two lexicon-based classifiers (SentiStrength-SE and DEVA), for which retraining is not possible. To enable comparison with the within-platform benchmark [30], we replicate the former experimental setting. Specifically, we split each gold set into training (70%) and test (30%) sets by performing stratified sampling with scikit-learn.[3] We evaluate the performance of all tools on the held-out test sets. As for the supervised classifiers, we first use the training set to retrain them using the methods provided by each toolkit; then, their performance is assessed in a cross-platform setting, by using the test set from the other experimental datasets (e.g., we train on the 70% train set of Stack Overflow and test on the 30% test sets from Jira and GitHub). Furthermore, we run twice the train and test for Senti4SD, because the feature set of Senti4SD can be customized. As such, we also run the train/test steps by removing the keyword-based features, that is, the uni- and bi-grams Bag of Words (BoW). The reason behind this choice is to understand the

[3] https://scikit-learn.org/stable/index.html

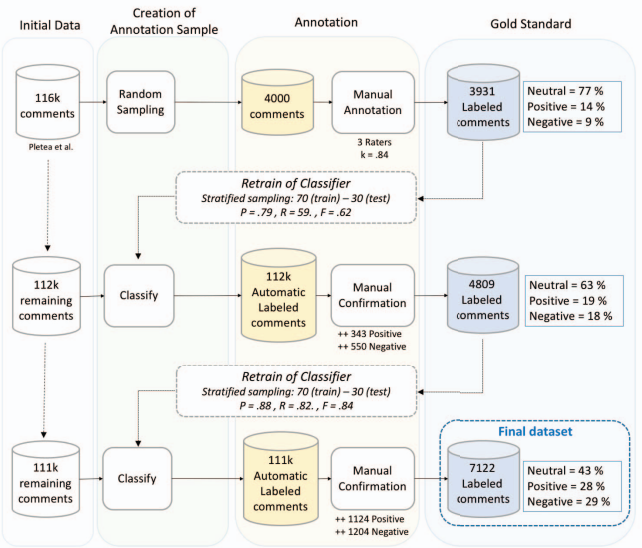


Figure 1: Creating the Gold Standard through manual annotation of polarity classes.

extent to which the interaction style, i.e., the specific lexicon or jargon observed in a given platform, has an impact on the performance. We cannot replicate this evaluation for SentiCR as it only exploits features based on BoW.

In this study, we aim to compare the results we achieved when training/testing supervised approaches on different datasets (cross-platform) with the performance observed when such approaches are trained and tested on the same dataset (within-platform), as done in our previous work [30]. However, even minor changes in the settings used in the two studies may lead to major differences in the results. To address this problem, we rerun the training/testing in the within-platform setting for comparison, following the approach we previously adopted and described in [30] and observed negligible differences in the tool performances. By doing so, all possible confounding factors are controlled, and we can be confident that the potential differences between the two scenarios (within- vs. cross-platform settings) would be due to the used training sets.

To address RQ3, we analyze the learning curves of the supervised tools in a within-platform setting. The goal of this evaluation is to identify the minimum size of the gold standard that makes re-training convenient for supervised tools as compared to using lexicon-based, non-customizable ones. The learning curves enable us to visually assess how the size of the training set influences the classification performance. We start by training the supervised tools using a subset of 5% of the original training set for each platform. At each step, we increment the training set size by 5%. At every iteration, the subset for training is extracted from scratch with stratified sampling, and the performance is assessed on the entire held-out 30% test set from the same platform. Given the unbalanced distribution of the Jira dataset, we repeated the performance of evaluation twice for Senti4SD, with and without performing data resampling. To enable comparison with SentiCR, we use SMOTE [10] by replicating the setting described by its authors [2].

Metrics. We report the performance of each sentiment analysis tool in terms of precision, recall, and F1-measure for all the three polarity classes. This choice is in line with previous research [19, 30] and is consistent with the standard methodology adopted for benchmarking of sentiment analysis systems as well as more general text categorization approaches in evaluations campaigns [9].

For the sake of completeness, we report the overall performance using both micro- and macro-averaging as aggregated metrics. Micro-averaging is known to be influenced by the performance on the majority class [41]. Conversely, the ability of a classifier to correctly identify items belonging to classes with few training instances is correctly assessed by the macro-average. Given the unbalanced distribution of the Jira dataset, in this study we rely on the macro-average, i.e., precision and recall are first evaluated locally for each class, and then globally by averaging the results of the different categories.

Furthermore, we use the weighted kappa (κ) [12, 49] to assess both the agreement with gold labels (RQ1) and the agreement among the three tools (RQ2). We distinguish between mild disagreement (weight = 1), i.e., the disagreement between negative/positive and neutral annotations, and strong disagreement (weight = 2), i.e., the disagreement between positive and negative judgments. We interpret κ as follows [49]: κ values less or equal to zero indicate that agreement is less than chance; the agreement is slight if $0.01 \leq \kappa \leq 0.20$, fair if $0.21 \leq \kappa \leq 0.40$, moderate if $0.41 \leq \kappa \leq 0.60$, substantial if $0.61 \leq \kappa \leq 0.80$ and almost perfect if $0.81 \leq \kappa \leq 1$. Both the weighted scheme and the interpretation of κ are the same adopted in the previous studies [19, 30].

6 RESULTS

6.1 Performance of SE-specific tools in cross-platform settings

RQ1 *- To what extent do different SE-specific sentiment analysis tools agree with the emotions of software developers when used as 'off-the-shelf' tools in a cross-platform setting?* In Table 2, we report the performance in the cross-platform setting of the four tools, both by polarity class and overall. In bold we highlight the best values for each metric. For the sake of comparison against the within-platform setting, we also report the performance obtained by replicating the our previous study [30] (reported in grey). For each dataset, we highlighted in Italic the differences with respect to the within-platform setting. Furthermore, we report the tool agreement with the manual labeling (see Table 3) in terms of both weighted Cohen κ and the percentage of cases in which each tool issues the correct prediction (perfect agreement with the gold label) as well as the percentage of wrong predictions (severe/mild disagreements).

In the cross-platform setting, we observe a drop in the performance of the supervised tools SentiCR and Senti4SD on all datasets, compared to the within-platform setting. Conversely to what is observed in the within-platform setting, the two lexicon-based tools outperform the supervised approaches when these are retrained in a cross-platform condition. Exceptions are the cross-platform setting with training performed on Stack Overflow and test on GitHub, where Senti4SD achieve the best performance (macro F1 =.82), and the setting with training performed on GitHub and test on Stack Overflow, where Senti4SD and SentiStrength-SE both achieve the

Table 2: Performance of SE-specific sentiment analysis tools in the cross-platform setting. For each setting, we highlight the best values for each metric in bold and the overall performance in Italic. The within-platform setting is reported in grey.

Setting	Train set	Polarity Class	Senti4SD			Senti4SD (no BoW)			SentiCR			SentiStrength-SE			DEVA		
			P	*R*	*F1*	*P*	*R*	*F1*	*P*	*R*	*F1*	*P*	*R*	*F1*	*P*	*R*	*F1*
Test set: GitHub																	
Within-platform	GitHub	Negative	.92	.90	.91	–			.90	.63	.74	.79	.77	.78	.65	.68	.67
		Neutral	.90	.93	.92				.76	.94	.84	.78	.86	.82	.83	.71	.77
		Positive	.95	.91	.93				.89	.85	.87	.86	.76	.81	.69	.81	.75
		Micro-avg.	*.92*	*.92*	*.92*				*.82*	*.82*	*.82*	*.80*	*.80*	*.80*	*.73*	*.73*	*.73*
		Macro-avg.	*.92*	*.92*	*.92*				*.85*	*.81*	*.82*	*.81*	*.80*	*.80*	*.72*	*.73*	*.73*
Cross-platform	Stack Overflow	Negative	**.79**	.50	.61	.75	**.83**	**.79**	.78	.34	.47	.79	.77	.78	.65	.68	.67
		Neutral	.71	.85	.77	.82	.78	.80	.60	**.93**	.73	.78	.86	**.82**	**.83**	.71	.77
		Positive	.76	.84	.80	**.88**	**.85**	**.86**	.86	.67	.75	.86	.76	.81	.69	.81	.75
		Micro-avg.	*.74*	*.74*	*.74*	***.82***	***.82***	***.82***	*.68*	*.68*	*.68*	*.80*	*.80*	*.80*	*.73*	*.73*	*.73*
		Macro-avg.	*.76*	***.84***	*.80*	***.82***	*.82*	***.82***	*.75*	*.65*	*.65*	*.81*	*.80*	*.80*	*.72*	*.73*	*.73*
Differences with the within-platform setting		*Micro-avg.*	*-.18*	*-.18*	*-.18*	*-.10*	*-.10*	*-.10*	*-.14*	*-.14*	*-.14*	–			–		
		Macro-avg.	*-.16*	*-.08*	*-.12*	*-.10*	*-.10*	*-.10*	*-.10*	*-.16*	*-.17*						
Cross-platform	Jira	Negative	.84	.51	.63	**.86**	.50	.64	.84	.24	.37	.79	**.77**	**.78**	.65	.68	.67
		Neutral	.59	.96	.73	.62	.95	.75	.51	**.98**	.67	.78	.86	**.82**	**.83**	.71	.77
		Positive	**.93**	.45	.61	.91	.57	.70	.92	.35	.51	.86	.76	**.81**	.69	**.81**	.75
		Micro-avg.	*.68*	*.68*	*.68*	*.71*	*.71*	*.71*	*.58*	*.58*	*.58*	***.80***	***.80***	***.80***	*.73*	*.73*	*.73*
		Macro-avg.	*.79*	*.64*	*.66*	*.79*	*.67*	*.69*	*.76*	*.52*	*.52*	***.81***	***.80***	***.80***	*.72*	*.73*	*.73*
Differences with the within-platform setting		*Micro-avg.*	*-.24*	*-.24*	*-.24*	*-.21*	*-.21*	*-.21*	*-.24*	*-.24*	*-.24*	–			–		
		Macro-avg.	*-.13*	*-.28*	*-.26*	*-.13*	*-.25*	*-.23*	*-.09*	*-.29*	*-.30*						
Test set: Stack Overflow																	
Within-platform	Stack Overflow	Negative	.81	.87	.84	–			.79	.74	.76	.74	.79	.76	.67	.79	.73
		Neutral	.86	.81	.84				.80	.82	.81	.76	.76	.76	.84	.68	.75
		Positive	.91	.92	.92				.88	.90	.89	.89	.84	.86	.85	.90	.87
		Micro-avg.	*.87*	*.87*	*.87*				*.83*	*.83*	*.83*	*.80*	*.80*	*.80*	*.79*	*.79*	*.79*
		Macro-avg.	*.86*	*.87*	*.87*				*.82*	*.82*	*.82*	*.80*	*.80*	*.79*	*.79*	*.79*	*.78*
Cross-platform	GitHub	Negative	.71	.72	.72	.71	**.80**	.75	.72	.41	.52	**.74**	.79	**.76**	.67	.79	.73
		Neutral	.71	.80	.75	.79	.75	**.77**	.64	**.88**	.74	.76	.76	.76	**.84**	.68	.75
		Positive	**.92**	.78	.84	.90	.85	**.87**	.84	.78	.81	.89	.84	.86	.85	**.90**	**.87**
		Micro-avg.	*.77*	*.77*	*.77*	***.80***	***.80***	***.80***	*.72*	*.72*	*.72*	***.80***	***.80***	***.80***	*.79*	*.79*	*.79*
		Macro-avg.	*.78*	*.77*	*.77*	***.80***	***.80***	***.80***	*.73*	*.69*	*.69*	***.80***	***.80***	*.79*	*.79*	*.79*	*.78*
Differences with the within-platform setting		*Micro-avg.*	*-.10*	*-.10*	*-.10*	*-.07*	*-.07*	*-.07*	*-.11*	*-.11*	*-.11*	–			–		
		Macro-avg.	*-.09*	*-.10*	*-.10*	*-.06*	*-.07*	*-.07*	*-.09*	*-.13*	*-.13*						
Cross-platform	Jira	Negative	.77	.34	.48	**.79**	.32	.46	.71	.13	.22	.74	**.79**	**.76**	.67	**.79**	.73
		Neutral	.56	.93	.70	.57	.93	.70	.46	**.97**	.62	.76	.76	**.76**	**.84**	.68	.75
		Positive	**.96**	.65	.78	.92	.70	.79	**.96**	.38	.54	.89	.84	.86	.85	**.90**	**.87**
		Micro-avg.	.68	.68	.68	.68	.68	.68	.54	.54	.54	**.80**	**.80**	**.80**	**.79**	.79	.79
		Macro-avg.	.76	.64	.65	.76	.65	.65	.71	.49	.46	**.80**	**.80**	.79	.79	.79	.78
Differences with the within-platform setting		*Micro-avg.*	*-.19*	*-.19*	*-.19*	*-.19*	*-.19*	*-.19*	*-.29*	*-.29*	*-.29*	–			–		
		Macro-avg.	*-.10*	*-.23*	*-.22*	*-.10*	*-.22*	*-.22*	*-.11*	*-.33*	*-.36*						
Test set: Jira																	
Within-platform	Jira	Negative	.75	.60	.67	–			.83	.63	.72	.64	.72	.68	.52	.69	.59
		Neutral	.87	.89	.88				.88	.91	.89	.93	.81	.87	.96	.75	.83
		Positive	.76	.78	.77				.79	.81	.80	.69	.93	.79	.63	.90	.74
		Micro-avg.	*.83*	*.83*	*.83*				*.86*	*.86*	*.86*	*.82*	*.82*	*.82*	*.77*	*.77*	*.77*
		Macro-avg.	*.79*	*.76*	*.77*				*.83*	*.78*	*.80*	*.75*	*.82*	*.78*	*.69*	*.78*	*.72*
Cross-platform	GitHub	Negative	.57	.64	.61	.57	.61	.59	**.75**	.56	.64	.64	**.72**	**.68**	.52	.69	.59
		Neutral	.89	.79	.84	.88	.80	.84	.90	**.87**	**.88**	.93	.81	.87	**.96**	.75	.83
		Positive	.65	.85	.74	.65	.81	.72	**.71**	.91	**.8**	.69	**.93**	.79	.63	.9	.74
		Micro-avg.	*.78*	*.78*	*.78*	*.78*	*.78*	*.78*	***.84***	***.84***	***.84***	*.82*	*.82*	*.82*	*.77*	*.77*	*.77*
		Macro-avg.	*.70*	*.76*	*.73*	*.70*	*.74*	*.71*	***.79***	*.78*	*.77*	*.75*	***.82***	***.78***	*.69*	*.78*	*.72*
Difference with the within-platform setting		*Micro-avg.*	*-.05*	*-.05*	*-.05*	*-.05*	*-.05*	*-.05*	*-.02*	*-.02*	*-.02*	–			–		
		Macro-avg.	*-.09*	*–*	*-.04*	*-.09*	*-.02*	*-.06*	*-.04*	*–*	*-.03*						
Cross-platform	Stack Overflow	Negative	.44	.26	.33	.50	.69	.58	.16	.03	.05	**.64**	**.72**	**.68**	.52	.69	.59
		Neutral	.83	.79	.81	.92	.73	.81	.75	**.94**	.83	.93	.81	**.87**	**.96**	.75	.83
		Positive	.57	.85	.68	.63	.90	.74	.72	.49	.58	**.69**	**.93**	**.79**	.63	.90	.74
		Micro-avg.	*.73*	*.73*	*.73*	*.76*	*.76*	*.76*	*.73*	*.73*	*.73*	***.82***	***.82***	***.82***	*.77*	*.77*	*.77*
		Macro-avg.	*.62*	*.63*	*.61*	*.68*	*.78*	*.71*	*.54*	*.49*	*.49*	***.75***	***.82***	***.78***	*.69*	*.78*	*.72*
Differences with the within-platform setting		*Micro-avg.*	*-.10*	*-.10*	*-.10*	*-.07*	*-.07*	*-.07*	*-.13*	*-.13*	*-.13*	–			–		
		Macro-avg.	*-.17*	*-.13*	*-.16*	*-.11*	*+.02*	*-.06*	*-.29*	*-.29*	*-.31*						

Table 3: Agreement of SE-specific tools with manual labelling. The within-platform setting for each dataset is reported in gray.

Train set	Classifier	Agreement metrics			
		k	*Perfect Agreement*	*Disagreement Severe*	*Disagreement Mild*
Test set: GitHub					
none	Senti-Strength-SE	.71	80%	4%	16%
	DEVA	.58	73%	9%	18%
Stack Overflow	Senti4SD	.61	74%	5%	21%
	SentiCR	.53	68%	3%	29%
Jira	Senti4SD	.52	68%	2%	30%
	SentiCR	.35	58%	1%	41%
GitHub	Senti4SD	.88	91%	1%	8%
	SentiCR	.74	83%	2%	15%
Test set: Stack Overflow					
none	Senti-Strength-SE	.74	80%	2%	18%
	DEVA	.71	79%	4%	17%
GitHub	Senti4SD	.69	77%	3%	20%
	SentiCR	.58	72%	5%	23%
Jira	Senti4SD	.55	68%	1%	31%
	SentiCR	.31	54%	1%	45%
Stack Overflow	Senti4SD	.83	87%	1%	12%
	SentiCR	.76	82%	3%	15%
Test set: Jira					
none	Senti-Strength-SE	.69	82%	1%	17%
	DEVA	.6	77%	2%	21%
GitHub	Senti4SD	.61	78%	1%	21%
	SentiCR	.68	84%	1%	16%
Stack Overflow	Senti4SD	.47	73%	2%	25%
	SentiCR	.33	73%	2%	25%
Jira	Senti4SD	.68	83%	–	17%
	SentiCR	.72	86%	–	14%

best performance (macro F1 = .80). The highest drop in performance is observed for SentiCR when Jira is used for training and GitHub (macro F1 = .52, representing a drop of 30% with respect to the within-platform setting) and Stack Overflow for testing (macro F1 = .46, indicating a drop of 36%). As a further confirmation of the results in Table 2, we observe a substantial agreement with the manual annotation for SentiStrength-SE (see Table 3). Conversely, the κ values in the cross-platform setting indicate a moderate to substantial agreement for Senti4SD and DEVA, and a fair to moderate agreement for SentiCR.

As for Senti4SD, a slight increase in performance is reported when BoW is excluded from the feature set in most settings (see Table 2). For the GitHub test set, the macro F1 of Senti4SD raises from .80 (with BoW) to .82 (without BoW) when training on Stack Overflow, and from .66 (with BoW) to .69 (without BoW) when training on Jira. We observe similar results for Stack Overflow when GitHub is used to train and for the Jira test set with training on Stack Overflow. Consistently, we observe the highest drop in macro-average from the within- to the cross-platform setting for SentiCR (-30%, -36%, and -31% decrease in macro F1 for the GitHub, Stack Overflow, and Jira test sets, respectively), which exploits a fixed feature set composed on uni- and bi-grams. This provides evidence of the lower ability to generalize of BoW features in cross-dataset settings, thus confirming the concerns of the NLP community about the risk of overfitting of model relying on n-gram features [20]. Looking at the performance of each polarity class, we observe that the drop in performance is mainly due to a drop in precision for the neutral class and recall for the negative and positive classes. This evidence suggests that positive and negative lexicon might be platform-dependent and, therefore, we lose recall for the non-neutral classes in cross-platform settings. This also reflects in the mild disagreement (i.e., the confounding between the positive and neutral, or between the negative and neutral classes) being the main cause of disagreement. Conversely, severe disagreement occurs at most in the 9% of cases, for DEVA on GitHub (see Table 3).

RQ2 *- To what extent do results from different SE-specific sentiment analysis tools agree with each other when used as 'off-the-shelf' tools in a cross-platform setting?* - In Table 4, we report the paired comparisons, using the same measures of agreement between each pair of tools. SentiStrength-SE and DEVA also show a substantial to almost perfect agreement with each other, ranging from $\kappa = .65$ for GitHub to $\kappa = .79$ for Stack Overflow, and $\kappa = .81$ for Jira. This is somewhat expected, considering that they share the same lexical resources and approach for polarity classification [17, 18]. The lowest agreement scores are observed for the lexicon-based tools and SentiCR, which is purely based on BoW. Senti4SD is in the middle of this scale, showing a moderate to substantial agreement with lexicon-based tools, probably because it relies on both lexicon-based features and BoW.

6.2 Error Analysis

We complement the quantitative analysis with a content analysis aimed at assessing the main causes of misclassification. We randomly sampled a subset of 320 texts (statistically significant sample size at 95% confidence level) from the documents for which both supervised classifiers yield a wrong prediction. Two of the authors independently labeled half of the cases and assigned a label choosing among the error categories identified in our previous benchmark study [30] (see Table 5). Then, they jointly discussed all cases to confirm the error labels. The goal of this analysis is to assess if the open challenges in sentiment analysis of developers' communication traces in a cross-platform setting are the same highlighted in a within-platform condition.

We found that the main cause of misclassification are *general errors*, occurring 68% of times. Such errors are caused by the inability of the tools to correctly deal with some textual cues. In most cases, this is due to lexical cues that are not recognized as either positive or negative because they do not occur frequently enough in the train set in order to hold sufficient predictive power. A special case is emoticons, which may have platform-dependent representation (e.g., ":smiley:" vs. ":-)"). General errors also occur due to wrong preprocessing (e.g., emoticons erroneously treated as non-unique tokens and rather split into its constituent characters), wrong spelling of words, or wrong negation handling.

The second cause for misclassification is the *subjectivity in sentiment annotation* (11%). Sentiment labeling is an inherently subjective task: even in the presence of clear annotation guidelines,

Table 4: Agreement of SE-specific tools with each other in cross-platform settings. The within-platform setting for each dataset is reported in gray.

Train set	Classifier	Agreement metrics			
		k	*Perfect Agreement*	*Disagreement Severe*	*Disagreement Mild*
Test set: GitHub					
–	SentiStrength-SE vs DEVA	0.65	78%	7%	15%
Stack Overflow	Senti4SD vs. SentiCR	0.48	68%	3%	29%
	Senti4SD vs. SentiStrength-SE	0.58	73%	5%	22%
	Senti4SD vs. DEVA	0.46	65%	8%	27%
	SentiCR vs. SentiStrength-SE	0.51	68%	3%	29%
	SentiCR vs. DEVA	0.47	64%	5%	31%
Jira	Senti4SD vs. SentiCR	0.49	78%	0%	22%
	Senti4SD vs. SentiStrength-SE	0.51	69%	2%	29%
	Senti4SD vs. DEVA	0.38	58%	4%	38%
	SentiCR vs. SentiStrength-SE	0.39	63%	1%	36%
	SentiCR vs. DEVA	0.30	53%	2%	45%
GitHub	Senti4SD vs. SentiCR	0.75	83%	2%	15%
	Senti4SD vs. SentiStrength-SE	0.71	81%	4%	15%
	Senti4SD vs. DEVA	0.59	73%	8%	19%
	SentiCR vs. SentiStrength-SE	0.65	77%	4%	19%
	SentiCR vs. DEVA	0.56	72%	7%	21%
Test set: Stack Overflow					
–	SentiStrength-SE vs DEVA	0.79	85%	4%	11%
GitHub	Senti4SD vs. SentiCR	0.59	73%	5%	22%
	Senti4SD vs. SentiStrength-SE	0.69	76%	2%	22%
	Senti4SD vs. DEVA	0.64	74%	5%	21%
	SentiCR vs. SentiStrength-SE	0.57	70%	4%	26%
	SentiCR vs. DEVA	0.55	69%	7%	24%
Jira	Senti4SD vs. SentiCR	0.44	74%	1%	25%
	Senti4SD vs. SentiStrength-SE	0.53	65%	1%	34%
	Senti4SD vs. DEVA	0.49	62%	2%	36%
	SentiCR vs. SentiStrength-SE	0.33	55%	1%	44%
	SentiCR vs. DEVA	0.29	49%	1%	50%
Stack Overflow	Senti4SD vs. SentiCR	0.75	82%	4%	14%
	Senti4SD vs. SentiStrength-SE	0.79	83%	2%	15%
	Senti4SD vs. DEVA	0.73	80%	5%	15%
	SentiCR vs. SentiStrength-SE	0.72	80%	4%	16%
	SentiCR vs. DEVA	0.68	79%	7%	14%
Test set: Jira					
–	SentiStrength-SE vs DEVA	0.81	90%	3%	7%
GitHub	Senti4SD vs. SentiCR	0.71	84%	1%	15%
	Senti4SD vs. SentiStrength-SE	0.71	83%	2%	15%
	Senti4SD vs. DEVA	0.63	79%	3%	18%
	SentiCR vs. SentiStrength-SE	0.76	87%	2%	11%
	SentiCR vs. DEVA	0.69	83%	3%	14%
Stack Overflow	Senti4SD vs. SentiCR	0.38	74%	1%	24%
	Senti4SD vs. SentiStrength-SE	0.61	79%	3%	18%
	Senti4SD vs. DEVA	0.54	75%	4%	21%
	SentiCR vs. SentiStrength-SE	0.33	69%	2%	29%
	SentiCR vs. DEVA	0.29	65%	3%	32%
Jira	Senti4SD vs. SentiCR	0.77	89%	0%	11%
	Senti4SD vs. SentiStrength-SE	0.70	84%	1%	15%
	Senti4SD vs. DEVA	0.63	79%	2%	19%
	SentiCR vs. SentiStrength-SE	0.76	87%	1%	12%
	SentiCR vs. DEVA	0.69	82%	1%	17%

the label assigned to a given text might be influenced by the personality traits of the human annotator [40]. In line with previous results [30], we observe that in some cases, the raters are conservative and provide a neutral label for mild expressions of emotions or opinions.

Furthermore, the specific research goal and applications of sentiment analysis might be another driver for labeling decisions. It is the case of *polar facts*, which are inherently desirable or undesirable facts, such as code patch acceptance (e.g., "fixed") or bug reports (e.g., "seems to be failing for a different reason now"), expressed with a neutral sentiment. Polar facts are the third cause of

Table 5: Distribution of error categories

Error category	#cases (%)
General error	214 (68%)
Subjectivity in annotation	35 (11%)
Polar facts	25 (8%)
Politeness	19 (6%)
Implicit sentiment polarity	16 (5%)
Figurative language	6 (2%)
Pragmatics	6 (2%)
Overall	320

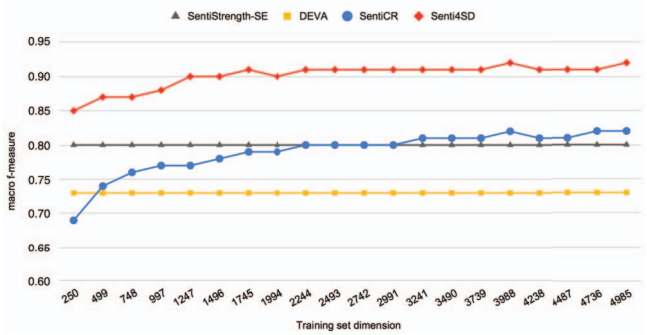
misclassification in the cross-platform setting (8%), as they might be inconsistently labeled across datasets, in line with the specific goals of the authors. For example, polar facts are often labeled as non-neutral in Jira. As an example, sentences such as "This seems to be failing for different reasons" or "This might be a bug indeed" are labeled as negative even if a neutral style is used (absence of emotions), probably due to the original intention of the authors of the Jira dataset to analyze the role of sentiment in issue tracking and its correlation with issue fixing time [27]. Polar facts are reported as the main cause of error in the within-platform setting [30].

The misclassification of sentences conveying *politeness* is a cause of error in 6% of cases, due to politeness expression such as "Thanks!" or "Sorry for" being inconsistently labeled across-dataset. As an example, in the Stack Overflow and GitHub datasets, politeness is considered neutral unless a clear expression of emotion is present in the text. This choice is in line with the evidence provided by computational linguists that emotion lexicon can be used for politeness expressions. This is typical of the so-called *behabitives* speech acts [3], in which no real feelings are expressed, but still emotional words are employed to convey other communicative intentions (e.g., "I am afraid this does not work"). As for Jira, thanking expression receive a positive label when they are related to code change approval (e.g., "thanks for the patch" is positive) indicating that positive polar facts receive a positive label (the patch is satisfying), while expression of gratitude (as in "Thanks!") are usually interpreted as neutral. Again, this is in line with the intention of Murgia et al. to study how sentiment correlates with issue-fixing time [27].

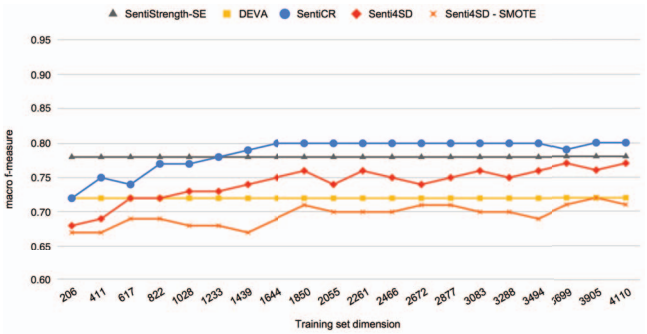
In 5% of cases, the sentiment is conveyed through indirect lexicon (*Implicit sentiment polarity*). As such, these comments are erroneously classified as neutral due to the absence of explicit lexical cues of sentiment. Finally, a few cases (2%) are misclassified due to the inability of the classifiers to deal with *figurative language*, as in the presence of humor or irony. The remaining 2% of cases are misclassified because the classifiers are not designed to take into account *pragmatics*. It is the case of questions or sentences reporting third persons' opinions or emotions, which are correctly labeled as neutral by humans but misclassified by the tool as positive or negative due to the presence of emotion words.

6.3 Learning curves for supervised classifiers

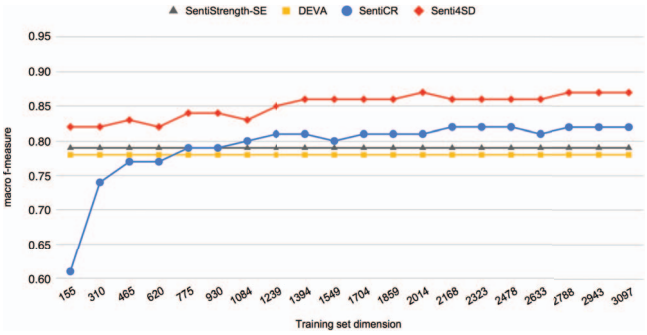
RQ3 *- To what extent is the performance of SE-specific sentiment analysis tools affected by the size of the training set?* - We want to assess how many documents we need to reliably retrain a supervised classifier for sentiment analysis in the software engineering domain. Accordingly, we analyze the learning curves of Senti4SD



(a) GitHub



(b) Jira



(c) Stack Overflow

Figure 2: Learning curves for the supervised tools evaluated in a within-platform setting using the GitHub (a), Jira (b), and Stack Overflow (c) datasets.

and SentiCR in a within-platform setting (see Figure 2). The performance of the lexicon-based tools, which cannot be customized, is reported for reference. We obtain the learning curves by plotting the performance on the test set of models created with training subsets of incremental size. We start by randomly sampling a subset of 5% of the original train set, using stratified sampling to preserve the polarity label distribution. At each iteration, we increase the training set at a rate of 5% and assess the model performance on the same 30% held-out test set used to address RQ1 and RQ2.

We observe that for GitHub and Stack Overflow, retraining Senti4SD is always convenient, even with a minimal set of documents, compared to the performance of lexicon-based tools. The nearly-optimal performance is obtained, for both datasets, with a train set of about 1,200 documents. A different situation is observed for Jira, which is largely unbalanced in favor of the neutral class (67% of the dataset). In this case, retraining is beneficial only if a larger set of documents is available (about 1,600 texts). For example, for SentiCR more than 1,200 documents are required to outperform SentiStrength-SE (see Figure 2.b). However, the improvement is negligible if compared to the one observed for GitHub (see Figure 2.a) and Stack Overflow (see Figure 2.c). A possible explanation for these results is that SentiStrength-SE was originally optimized using a subset of the Jira gold standard as a reference [17], which may arguably explain its very good performance on it. Another possible explanation for this difference in the performance could reside in the Jira dataset being unbalanced, thus making retraining not as effective as for GitHub and Stack Overflow, which are well-balanced datasets. As such, we included an additional setting for Jira where we performed class-balancing using SMOTE also for Senti4SD (SMOTE is the default preprocessing for SentiCR). This evidence suggests that even if resampling is performed before retraining, SentiStrength-SE still outperforms the other tools. As a further possible explanation, we hypothesize that the quality of the gold standard, measured in terms of inter-rater agreement, is also a major fact influencing the quality and reliability of the learned classification model. In fact, for both GitHub and Stack Overflow, κ values indicate a substantial to almost perfect agreement, while a lower agreement is observed for Jira (see Section 4).

7 DISCUSSION

In the following, we derive empirically-driven guidelines for reliable sentiment analysis in SE, based on the findings of the current study.

Perform SE-specific tuning for enhanced accuracy. Domain adaptation is a well-known problem in machine learning [4], in general, and in sentiment analysis, in particular [38]. Our previous benchmarking study performed in a within-platform setting on the Stack Overflow and Jira datasets demonstrated that SE-specific tuning is beneficial for ensuring reliable sentiment analysis on technical texts [30]. We confirm these findings also on the GitHub dataset that we developed for the purpose of enriching the benchmark in the current study. In particular, we report comparable performance for the lexicon-based tools SentiStrength-SE and DEVA, thus providing further evidence that reliable sentiment analysis in software engineering is a feasible task.

Perform platform-specific tuning. The results of our benchmark study demonstrate how retraining across platforms does not work well for supervised tools, thus suggesting that the definition of 'domain' might be even narrowed-down at the level of the specific platform. In fact, despite our benchmark included only SE-specific datasets, we observe a drop in performance when supervised models are trained and tested on data gathered from different collaborative development environments. This suggests that semantics shifts also occur due to platform-specific jargon and communication style. In line with this evidence, we report better performance in the absence of BoW-based features (i.e., for Senti4SD no BoW, see Table 2) indicating the lower ability of n-grams to generalize, i.e., they might cause overfitting to the platform-specific lexicon, thus negatively

affecting the performance of supervised tools. This is further confirmed by the results of our error analysis (see Section 6.2). As such, whenever a gold standard is available, we recommend platform-specific retraining to enable correct modeling of the interaction style and lexicon of the specific platform.

Build a robust gold standard. In building a gold standard, one open issue is the correct amount of data required for retraining a reliable supervised classifier. To address this question, we performed a within-platform study and built the learning curves obtained with training sets of incremental size. The results, depicted in Figure 2, show that learning from unbalanced, low-agreement data might produce unsatisfying results even in a within-platform setting. This claim is in line with previous findings suggesting that the quality [1, 46] and internal consistency [30] of gold standards are crucial properties for successful training of classifiers.

Select the appropriate tool in line with the research goals. In the absence of a platform-specific gold standard for retraining, unsupervised tools or 'off-the-shelf' use of supervised classifiers are the only possible options. In both cases, we recommend using a tool only if a preliminary sanity check produces satisfying results on the target platform. Specifically, we recommend to collect and manually annotate sample data from the target platform in order to verify the alignment between the classification output and the manually-provided labels. Indeed, one of the most dangerous assumptions when reusing sentiment analysis tools and datasets is assuming agreement with the goals and sentiment conceptualization as originally thought by their authors. Our error analysis shows that even when sharing the theoretical model of emotion (e.g., the Shaver model used for the three datasets), the human raters may provide polarity labels based on their subjective perception or the specific research goals. It is the case of politeness, which is labeled inconsistently across datasets (see Section 6.2), thus inducing misclassification in the cross-platform settings.

8 THREATS TO VALIDITY

We are aware that the methodology adopted could produce different results if applied to different datasets and, therefore, that the choice of datasets to include in the benchmark might represent a threat to conclusion validity. As such, we included all the model-driven gold standards for sentiment annotation in software engineering that are available at the time of writing, composed of posts (questions, answers, and comments) from Stack Overflow and comments from Jira. To further mitigate this threat, we built a third gold standard dataset including comments from GitHub.

All datasets in our benchmark are built by collecting documents from platforms that are popular and widely adopted among software developers. As such, we included three major collaborative software development platforms. Each platform supports different collaborative tasks, from technical question-answering (Stack Overflow) to issue tracking (Jira), to collaborative software development with version control (GitHub). Given the dataset size and the variety of tasks considered, we are reasonably confident that the datasets included in this study are representative of the developers' communication, thus reducing threats to external validity.

A threat to construct validity is due to sentiment analysis being inherently affected by the subjectivity of the studied phenomenon, i.e., emotions and opinions as conveyed in text [40]. In our previous research [30], we showed how model-driven annotation is crucial to obtain a high-quality, reliable gold standard for training emotion polarity classifiers. Inconsistency in the annotation guidelines might be a cause of a drop in performance *per se*. As such, we addressed this threat by including in our benchmark only model-driven datasets. Furthermore, the GitHub dataset, which we built from scratch, is annotated following the same guidelines and adopting the same theoretical model of emotions leveraged for creating the Stack Overflow and Jira gold standards. This choice reduces the risk of confounding factors due to different annotation schema, thus enabling us to correctly assess the impact of the cross-platform train-test condition.

Finally, threats to internal validity concern internal factors such as the configuration of the parameters for the machine learning algorithms implemented by Senti4SD and SentiCR. To mitigate this threat, we replicated the experimental conditions under which the tools were originally validated [6], [2], using the available training toolkits. Furthermore, we ran again the within-platform setting to enable a fair comparison with the results reported in our previous research [30].

9 CONCLUSIONS

In this paper, we assessed the performance of four available SE-specific sentiment analysis tools in a cross-platform setting. We found that the retraining of SE-specific sentiment analysis tools is not a viable solution when the training and test sets come from different data sources. Conversely, better performance is observed for lexicon-based approaches, which we recommend whenever retraining is not possible due to the unavailability of a gold standard. However, further evidence shows that supervised tools achieve better performance than lexicon-based ones when retrained with a minimal training set of about 1,000 documents, as long as the training set is balanced and substantial inter-rater agreement is observed. Based on our empirical findings, we derived guidelines for reliable sentiment analysis in software engineering. Finally, we built a dataset of over 7,000 manually annotated GitHub comments, which we release to support future studies in the field.

In future work, we plan to further enhance the understanding of classification performance drop under domain- and platform-shift, by including the assessment of predictive power of features across additional datasets. Also, we plan to assess the cross-platform performance of approaches based on deep learning, which are not included in this study.

10 ACKNOWLEDGMENTS

We thank Giovanna Saracino for contributing to the early stage of this study.

REFERENCES

[1] Amritanshu Agrawal and Tim Menzies. 2018. Is "Better Data" Better than "Better Data Miners"? On the Benefits of Tuning SMOTE for Defect Prediction *(ICSE '18)*. ACM, New York, NY, USA, 1050–1061. https://doi.org/10.1145/3180155.3180197

[2] T. Ahmed, A. Bosu, A. Iqbal, and S. Rahimi. 2017. SentiCR: A customized sentiment analysis tool for code review interactions. In *2017 32nd IEEE/ACM International Conf. on Automated Software Engineering (ASE)*. IEEE Press, 106–111. https://doi.org/10.1109/ASE.2017.8115623

[3] John L. Austin. 1962. *How to do things with words*. Oxford University Press.

[4] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. 2010. A theory of learning from different domains. *Machine Learning* 79, 1 (2010), 151–175. https://doi.org/10.1007/s10994-009-5152-4

[5] Cássio Castaldi Araujo Blaz and Karin Becker. 2016. Sentiment Analysis in Tickets for IT Support *(MSR '16)*. ACM, New York, NY, USA, 235–246. https://doi.org/10.1145/2901739.2901781

[6] Fabio Calefato, Filippo Lanubile, Federico Maiorano, and Nicole Novielli. 2018. Sentiment Polarity Detection for Software Development. *Empirical Software Engineering* 23, 3 (2018), 1352–1382. https://doi.org/10.1007/s10664-017-9546-9

[7] Fabio Calefato, Filippo Lanubile, and Nicole Novielli. 2018. How to ask for technical help? Evidence-based guidelines for writing questions on Stack Overflow. *Information & Software Technology* 94 (2018), 186–207. https://doi.org/10.1016/j.infsof.2017.10.009

[8] Fabio Calefato, Filippo Lanubile, Nicole Novielli, and Luigi Quaranta. 2019. EMTk: The Emotion Mining Toolkit *(SEmotion '19)*. IEEE Press, 34–37. https://doi.org/10.1109/SEmotion.2019.00014

[9] Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso. 2018. Evalita 2018: Overview on the 6th Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. In *Proc. of the Sixth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2018) co-located with the Fifth Italian Conf. on Computational Linguistics (CLiC-it 2018), Turin, Italy, December 12-13, 2018*. CEUR-SW.org. http://ceur-ws.org/Vol-2263/paper001.pdf

[10] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. 2002. SMOTE: Synthetic Minority Over-sampling Technique. *J. Artif. Intell. Res.* 16 (2002), 321–357. https://doi.org/10.1613/jair.953

[11] Zhenpeng Chen, Yanbin Cao, Xuan Lu, Qiaozhu Mei, and Xuanzhe Liu. 2019. SEntiMoji: An Emoji-Powered Learning Approach for Sentiment Analysis in Software Engineering *(ESEC/FSE 2019)*. ACM, New York, NY, USA, 841–852. https://doi.org/10.1145/3338906.3338977

[12] Jacob Cohen. 1968. Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological Bulletin* 70, 4 (1968), 213. https://doi.org/10.1037/h0026256

[13] Daviti Gachechiladze, Filippo Lanubile, Nicole Novielli, and Alexander Serebrenik. 2017. Anger and Its Direction in Collaborative Software Development *(ICSE-NIER '17)*. IEEE Press, 11–14. https://doi.org/10.1109/ICSE-NIER.2017.18

[14] Emitza Guzman, Rana Alkadhi, and Norbert Seyff. 2016. A Needle in a Haystack: What Do Twitter Users Say about Software?. In *24th IEEE International Requirements Engineering Conf., RE 2016, Beijing, China, September 12-16, 2016*. IEEE, 96–105. https://doi.org/10.1109/RE.2016.67

[15] Emitza Guzman, David Azócar, and Yang Li. 2014. Sentiment Analysis of Commit Comments in GitHub: An Empirical Study *(MSR 2014)*. ACM, New York, NY, USA, 352–355. https://doi.org/10.1145/2597073.2597118

[16] H. He and E. A. Garcia. 2009. Learning from Imbalanced Data. *IEEE Transactions on Knowledge and Data Engineering* 21, 9 (2009), 1263–1284. https://doi.org/10.1109/TKDE.2008.239

[17] Md Rakibul Islam and Minhaz F. Zibran. 2017. Leveraging Automated Sentiment Analysis in Software Engineering *(MSR '17)*. IEEE Press, 203–214. https://doi.org/10.1109/MSR.2017.9

[18] Md Rakibul Islam and Minhaz F. Zibran. 2018. DEVA: sensing emotions in the valence arousal space in software engineering text. In *Proc. of the 33rd Annual ACM Symposium on Applied Computing, SAC 2018, Pau, France, April 09-13, 2018*. 1536–1543. https://doi.org/10.1145/3167132.3167296

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

[20] Daniel Jurafsky and James H. Martin. 2009. *Speech and Language Processing (2nd Edition)*. Prentice-Hall, Inc., USA.

[21] Richard S. Lazarus. 1991. *Emotion and adaptation / Richard S. Lazarus.* Oxford University Press New York. xiii, 557 p. ; pages. http://www.loc.gov/catdir/enhancements/fy0602/91009611-t.html

[22] Bin Lin, Fiorella Zampetti, Gabriele Bavota, Massimiliano Di Penta, and Michele Lanza. 2019. Pattern-Based Mining of Opinions in Q&A Websites *(ICSE '19)*. IEEE Press, 548–559. https://doi.org/10.1109/ICSE.2019.00066

[23] 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? *(ICSE '18)*. ACM, New York, NY, USA, 94–104. https://doi.org/10.1145/3180155.3180195

[24] Walid Maalej, Zijad Kurtanoviundefined, Hadeer Nabil, and Christoph Stanik. 2016. On the Automatic Classification of App Reviews. *Requir. Eng.* 21, 3, 311–331. https://doi.org/10.1007/s00766-016-0251-9

[25] Mika Mäntylä, Bram Adams, Giuseppe Destefanis, Daniel Graziotin, and Marco Ortu. 2016. Mining Valence, Arousal, and Dominance: Possibilities for Detecting Burnout and Productivity? *(MSR '16)*. ACM, New York, NY, USA, 247–258. https://doi.org/10.1145/2901739.2901752

[26] T. Menzies. 2020. The Five Laws of SE for AI. *IEEE Software* 37, 1 (Jan 2020), 81–85. https://doi.org/10.1109/MS.2019.2954841

[27] Alessandro Murgia, Parastou Tourani, Bram Adams, and Marco Ortu. 2014. Do Developers Feel Emotions? An Exploratory Analysis of Emotions in Software Artifacts *(MSR 2014)*. ACM, New York, NY, USA, 262–271. https://doi.org/10.1145/2597073.2597086

[28] Nicole Novielli, Andrew Begel, and Walid Maalej. 2019. Introduction to the special issue on affect awareness in software engineering. *Journal of Systems and Software* 148 (2019), 180 – 182. https://doi.org/10.1016/j.jss.2018.11.016

[29] Nicole Novielli, Fabio Calefato, and Filippo Lanubile. 2015. The Challenges of Sentiment Detection in the Social Programmer Ecosystem *(SSE 2015)*. ACM, New York, NY, USA, 33–40. https://doi.org/10.1145/2804381.2804387

[30] Nicole Novielli, Daniela Girardi, and Filippo Lanubile. 2018. A Benchmark Study on Sentiment Analysis for Software Engineering Research *(MSR '18)*. ACM, New York, NY, USA, 364–375. https://doi.org/10.1145/3196398.3196403

[31] N. Novielli and A. Serebrenik. 2019. Sentiment and Emotion in Software Engineering. *IEEE Software* 36, 5 (2019), 6–23. https://doi.org/10.1109/MS.2019.2924013

[32] 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 *(MSR '15)*. IEEE Press, 303–313.

[33] Marco Ortu, Alessandro Murgia, Giuseppe Destefanis, Parastou Tourani, Roberto Tonelli, Michele Marchesi, and Bram Adams. 2016. The Emotional Side of Software Developers in JIRA *(MSR '16)*. ACM, New York, NY, USA, 480–483. https://doi.org/10.1145/2901739.2903505

[34] Bo Pang and Lillian Lee. 2008. Opinion Mining and Sentiment Analysis. *Foundations and Trends in Information Retrieval* 2, 1-2 (2008), 1–135. https://doi.org/10.1561/1500000011

[35] Sebastiano Panichella, Andrea Di Sorbo, Emitza Guzman, Corrado A. Visaggio, Gerardo Canfora, and Harald C. Gall. 2015. How Can i Improve My App? Classifying User Reviews for Software Maintenance and Evolution. In *Proc. of the 2015 IEEE International Conf. on Software Maintenance and Evolution (ICSME) (ICSME '15)*. IEEE Computer Society, USA, 281–290. https://doi.org/10.1109/ICSM.2015.7332474

[36] Daniel Pletea, Bogdan Vasilescu, and Alexander Serebrenik. 2014. Security and Emotion: Sentiment Analysis of Security Discussions on GitHub *(MSR 2014)*. ACM, New York, NY, USA, 348–351. https://doi.org/10.1145/2597073.2597117

[37] Ellen Riloff, Siddharth Patwardhan, and Janyce Wiebe. 2006. Feature Subsumption for Opinion Analysis *(EMNLP '06)*. ACL, USA, 440–448.

[38] Sebastian Ruder and Barbara Plank. 2018. Strong Baselines for Neural Semi-Supervised Learning under Domain Shift. In *Proc. of the 56th Annual Meeting of the Association for Computational Linguistics (Vol. 1: Long Papers)*. ACL, 1044–1054. https://doi.org/10.18653/v1/P18-1096

[39] J.A. Russell. 1980. A circumplex model of affect. *Journal of personality and social psychology* 39, 6 (1980), 1161–1178.

[40] Klaus R. Scherer, Tanja Wranik, Janique Sangsue, Véronique Tran, and Ursula Scherer. 2004. Emotions in everyday life: probability of occurrence, risk factors, appraisal and reaction patterns. *Social Science Information* 43, 4 (2004), 499–570. https://doi.org/10.1177/0539018404047701

[41] Fabrizio Sebastiani. 2002. Machine learning in automated text categorization. *ACM Comput. Surv.* 34, 1 (2002), 1–47. https://doi.org/10.1145/505282.505283

[42] Phillip Shaver, Judith Schwartz, Donald Kirson, and O'Connor Cary. 1987. Emotion knowledge: Further exploration of a prototype approach. *Journal of Personality and Social Psychology* 52, 6 (1987), 1061—-1086. https://doi.org/10.1037/0022-3514.52.6.1061

[43] Vinayak Sinha, Alina Lazar, and Bonita Sharif. 2016. Analyzing Developer Sentiment in Commit Logs *(MSR '16)*. ACM, New York, NY, USA, 520–523. https://doi.org/10.1145/2901739.2903501

[44] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proc. of the 2013 Conf. on Empirical Methods in Natural Language Processing*. ACL, Seattle, Washington, USA, 1631–1642. https://www.aclweb.org/anthology/D13-1170

[45] Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. 2011. Lexicon-Based Methods for Sentiment Analysis. *Comput. Linguist.* 37, 2 (June 2011), 267–307. https://doi.org/10.1162/COLI_a_00049

[46] Chakkrit Tantithamthavorn, Shane McIntosh, Ahmed E. Hassan, Akinori Ihara, and Kenichi Matsumoto. 2015. The Impact of Mislabelling on the Performance and Interpretation of Defect Prediction Models *(ICSE '15)*. IEEE Press, 812–823.

[47] Mike Thelwall, Kevan Buckley, Georgios Paltoglou, Di Cai, and Arvid Kappas. 2010. Sentiment Strength Detection in Short Informal Text. *J. Am. Soc. Inf. Sci. Technol.* 61, 12 (Dec. 2010), 2544–2558.

[48] Gias Uddin and Foutse Khomh. 2017. Opiner: An Opinion Search and Summarization Engine for APIs *(ASE 2017)*. IEEE Press, 978–983.

[49] Anthony Viera and Joanne Garrett. 2005. Understanding Interobserver Agreement: The Kappa Statistic. *Family medicine* 37 (06 2005), 360–3.

[50] Lei Zhang, Shuai Wang, and Bing Liu. 2018. Deep Learning for Sentiment Analysis: A Survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* (01 2018). https://doi.org/10.1002/widm.1253