Noname manuscript No.
(will be inserted by the editor)

Transformers and Meta-Tokenization in Sentiment Analysis for Software Engineering

Nathan Cassee · Andrei Agaronian · Eleni Constantinou · Nicole Novielli · Alexander Serebrenik

the date of receipt and acceptance should be inserted later

Abstract Sentiment analysis has been used to study aspects of software engineering, such as issue resolution, toxicity, and self-admitted technical debt. To address the peculiarities of software engineering texts, sentiment analysis tools often consider the specific technical lingo practitioners use. To further improve the application of sentiment analysis, there have been two recommendations: Using pre-trained transformer models to classify sentiment and replacing non-natural language elements with meta-tokens. In this work, we benchmark five different sentiment analysis tools (two pre-trained transformer models and three machine learning tools) on 2 gold-standard sentiment analysis datasets. We find that pre-trained transformers outperform the best machine learning tool on only one of the two datasets, and that even on that dataset the performance difference is a few percentage points. Therefore, we recommend that software engineering researchers should not just consider predictive performance when selecting a sentiment analysis tool because the best-performing sentiment analysis tools perform very similarly to each other (within 4 percentage points) . Meanwhile, we find that meta-tokenization does not improve

Nathan Cassee
Eindhoven University of Technology, The Netherlands
E-mail: n.w.cassee@tue.nl

Andrei Agaronian
Eindhoven University of Technology, The Netherlands
E-mail: andrei.agaronian@gmail.com

Eleni Constantinou
University of Cyprus, Cyprus
E-mail: constantinou.a.eleni@ucy.ac.cy

Nicole Novielli
University of Bari, Italy
E-mail: nicole.novielli@uniba.it

Alexander Serebrenik
Eindhoven University of Technology, The Netherlands
E-mail: a.serebrenik@tue.nl

the predictive performance of sentiment analysis tools. Both of our findings can be used by software engineering researchers who seek to apply sentiment analysis tools to software engineering data.

Keywords Sentiment Analysis, Sentiment Analysis for Software Engineering, meta-tokenization, Sentiment Analysis Tools

1 Introduction

The increasing complexity of modern software engineering projects has resulted in software engineering becoming an inherently collaborative process. To help developers understand and manage software projects researchers have studied emotions and sentiment in software engineering because expressions of negative sentiment in software engineering projects could be used to identify potential problems (Lin et al., 2022). For instance, while studying sentiment Calefato et al. (2018b) found that successful questions on StackOverflow are short, and more importantly, do not express any sentiment, negative or positive. In a similar vein, Lanovaz and Adams (2019) found that negative posts on the R mailing lists were less likely to be responded to. In addition to these topics, studies have also investigated sentiment expressed in software engineering artifacts such as code reviews (Ahmed et al., 2017; Bosu et al., 2015; Paul et al., 2019), questions asked by developers (Uddin and Khomh, 2021) and issues (Maalej and Nabil, 2015; Ortu et al., 2019). However, there are many areas of software engineering in which sentiment analysis can be expected to be beneficial but has not yet been applied (Lin et al., 2022). In this work, we define sentiment analysis as a classification task in which a piece of text is assigned to a polarity class (usually positive, negative or neutral).

On the meta-level researchers have also studied how one can effectively study sentiment in software engineering (Biswas et al., 2020; Chen et al., 2019; Jongeling et al., 2017; Novielli et al., 2020, 2021). These studies have resulted in several practical recommendations on how one should use sentiment analysis tools on software engineering data. In this paper, we are interested in two recent recommendations, and we seek to verify them. Through studying these recommendations we seek to further understand how software engineering researchers can more effectively study expressions of sentiment in software engineering.

The first recommendation we study in this paper originates from two studies of Biswas et al. (2020) and Chen et al. (2019) who recommend the usage of deep-learning-based sentiment analysis tools to classify sentiment in software engineering texts. However, contradicting the recommendations of Biswas *et al.* and Chen *et al.*, Lin et al. (2022) found that machine-learning approaches outperform deep-learning approaches when the size of the datasets is small. The exact reason for the misalignment between the recommendations of Biswas *et al.* and Chen *et al.* and the work of Lin *et al.* is not clear. One possible explanation might be related to different datasets being used in each of the benchmarks. Alternatively, the differences might be attributed to the appropriateness of the training of the machine-learning tools. For instance, Shwartz-Ziv and Armon (2022) found that deep-learning tools do not always outperform machine-learning tools. Fu and Menzies (2017), Pamungkas et al. (2020) and Yedida and Menzies (2022) studied similar questions in software engineering. They find that both machine-learning classifiers (such as SVM) and more simple deep-learning tools can outperform more complex deep-learners on various types of data. In this paper, we take the recommendations to use deep-learners to classify sentiment in software engineering texts (Biswas et al., 2020; Chen et al., 2019), and the work that finds that deep-learners do not always outperform non-deep-learning machine-learning tools (Fu and Menzies, 2017; Lin et al., 2018; Shwartz-Ziv and Armon, 2022; Yedida and Menzies, 2022). In a robust experimental set-up we seek to verify the existing recommendation, and we aim to understand how existing practices and recommendations can be updated to accurately apply sentiment analysis to software engineering data. Therefore, we pose:

RQ_1: Do existing deep-learning sentiment analysis models outperform machine-learning-based sentiment analysis tools?

The second recommendation we investigate in this work is the recommendation of Efstathiou and Spinellis (2018) to replace non-natural language in technical texts with tokens that capture the meaning of non-natural language. In this work we refer to this practice as *meta-tokenization*, however, this practice is also known as semantic categorization (Stanojevic and Vraneš, 2009). Text extracted from social coding platforms, such as GitHub, might contain different types of non-natural language elements like code-snippets, stacktraces and references to pull-requests. Several detection techniques for non-natural language in technical texts already exist: Such as NLoN (Mäntylä et al., 2018), or an approach authored by Bacchelli et al. (2010). Finally, Efstathiou and Spinellis (2018) proposes replacing these non-natural language elements that occur in code reviews with *meta-tokens*, where each meta-token replaces a specific type of non-natural language element. As existing sentiment analysis tools obtain performance scores of 90%, we seek to understand whether a consistent meta-tokenization approach further improves the performance of sentiment analysis tools. Therefore we pose:

RQ_2: How does the replacement of non-natural language elements in sentiment analysis data with meta-tokens affect the performance of Sentiment Analysis tools?

To study the two research-questions posed in this work we follow existing recommendations (Novielli et al., 2020) and we take two *gold-standard* datasets tailored for software engineering . We benchmark five state-of-the-art machine-learning and deep-learning sentiment analysis tools made for software engineering using these two datasets. To answer *RQ_1*, we take each tool and train it on a train split of the dataset and then evaluate the predictive performance of the tool on a test split of the same dataset. To ensure the validity of the results, we ensure the benchmarks are as robust as possible and we validate the recommendation by comparing the performance scores of the machine-learning and deep-learning-based tools.

To address *RQ2* we train sentiment-analysis tools on both the original version of the dataset, and a version of the dataset that has been processed such that non-natural language elements identified through a mix of manual and automated detection techniques have been replaced with meta-tokens. We then evaluate the predictive performance of each tool after training it on both versions of each dataset. And we test whether the predictive performance of the tool trained on the meta-tokenized version of the dataset is higher than on the tool trained on the original version of the dataset.

Based on the experiments we conduct for *RQ1* we find that there exists a small but observable performance differences between machine learners and deep learners. The best-performing machine learner, Senti4SD, outperforms one of the two evaluated deep-learning tools on one dataset. While on another dataset both deep learners outperform Senti4SD. However, while these performance differences exist they are minor, with performance scores differing by at most four percentage points. Meanwhile, for *RQ2* we find that meta-tokenization does not significantly improve the performance of any of the five sentiment analysis tools evaluated in this study.

Our work has several findings for researchers that aim to apply sentiment analysis tools to better understand software engineering :

– Predictive performance of deep-learning and machine-learning sentiment tools on gold-standard datasets is comparable: performance differences between tools do not exceed four percentage points.
– The presence of non-natural language elements in the current gold-standard datasets and the replacement of the non-natural language elements with meta-tokens does not significantly affect the performance of sentiment analysis tools.

This paper is structured as follows: Section 2 describes the used methodology, Section 3 lists the results, Section 4 argues that our chosen methodology is sound, Section 5 discusses the implications of our work, Section 6 discusses threats to validity, Section 7 discusses related work, and Section 8 concludes the paper.

2 Methodology

For this study we are interested in the performance of sentiment-analysis tools. To address *RQ1* we compare the performance of machine-learning tools with deep-learning-based tools. Additionally, we address *RQ2* by studying the performance of sentiment analysis tools after retraining them on a meta-tokenized version of a dataset.

2.1 Tools & Datasets

Datasets: In line with recent recommendations of Novielli et al. (2020) we select gold-standard sentiment analysis datasets. For this study, we define *gold-standard* as the largest and most rigorous datasets in the field of sentiment analysis for software engineering. In practice, this means the largest available balanced datasets have been labeled using theoretical models of affect by raters that achieve high inter-rater agreement (Novielli et al., 2020). For this study, we select the following gold-standard datasets:

– *Github gold-standard*: As GitHub is one of the most popular open-source platform, used by developers to work on collaborative software projects we select the gold-standard dataset authored by Novielli et al. (2020). This is a balanced dataset of 7,122 items, where 28%, 43% and 29% of posts convey negative, neutral and positive sentiment respectively. Each item in the dataset has been annotated by three authors using predefined annotation guidelines. The items in the dataset have been sampled from comments on commits and pull-requests taken from 90 GitHub repositories that were part of the 2014 MSR Challenge dataset. (Novielli et al., 2020)
– *StackOverflow gold-standard*: StackOverflow is a well-studied Q&A platform used by developers. The dataset of Calefato et al. (2018a) is a balanced dataset of 4,423 items ($\simeq 27\%$ negative, $\simeq 38\%$ neutral, $\simeq 35\%$ positive), labeled by several labelers that used predefined annotation guidelines. Additionally, the labelers of Calefato *et al.* achieved high inter-rater agreement. The items in the dataset have been sampled based on the presence of affective lexicons from a StackOverflow dump that covers the timeframe from July 2008 to September 2015. The sampled items are a combination of questions, answers, and comments. (Calefato et al., 2018a)

Tools: To find sentiment analysis tools for this study we use the list of tool identified by Lin et al. (2022). We select sentiment analysis tools that are publicly available, have been peer-reviewed, can be retrained, and have been designed for an application in Software Engineering. This has resulted in the list of the following four tools: SEntiMoji (Chen et al., 2019), SentiSW (Ding et al., 2018), SentiCR (Ahmed et al., 2017) and Senti4SD[1] (Calefato et al., 2018a). As the papers included in the literature study of Lin *et al.* have been gathered in 2019 it does not include the most recently released tools. Therefore, we also include a BERT-based transformer tuned for sentiment analysis published by Zhang et al. (2020).

SEntiMoji (Chen et al., 2019) is a deep-learning sentiment analysis tool based on a sentiment analysis tool that was originally designed for Twitter. It has been trained on Twitter, and is fine-tuned by the authors on Software Engineering data.

The *BERT-based transformers* published by Zhang *et al.* (Zhang et al., 2020) are deep-learning models that attempt to leverage existing large-scale

[1] Note that we used PySenti4SD, as this is the more recent version of Senti4SD.

language models to classify sentiment in software engineering text accurately. The pre-trained models are finetuned by the authors on Software Engineering datasets, and a comprehensive and re-usable replication package is available. In the paper, the authors evaluate four different pre-trained transformers. For this study we select one of the transformers that achieves competitive scores: *Bert*.

SentiSW (Ding et al., 2018) is built to classify the sentiment of issues comments on Github. The authors of SentiSW use a preprocessing pipeline to process the input and create TF-IDF vectors, finding that Gradient Boosting Tree (Pennacchiotti and Popescu, 2011) is the most accurate classifier.

SentiCR (Ahmed et al., 2017) is a sentiment analysis tool built to analyze code reviews on Github. It uses a preprocessing pipeline that performs operations such as the processing of negations and the generation of feature vectors based on TF-IDF. Finally, a Gradient Boosting Tree (Pennacchiotti and Popescu, 2011) is used to predict the sentiment. SentiCR as originally trained by the authors, is suited for binary classification: *Is negative sentiment present yes or no?* We retrain SentiCR using datasets containing both positive, negative, and neutral sentiments, and as such, we use it for ternary classification (positive, negative, or neutral). The only training parameter we modify is the oversampling of the minority item. The original authors use a value of 0.5, we set it to auto such that all classes except the majority class are resampled.

Senti4SD (Calefato et al., 2018a) uses a mix of lexicon-based, keyword-based, and semantic features to process input. Together with these features, the authors of Senti4SD use a word2vec (Mikolov et al., 2013) model and finally train a Support Vector Machine (Cortes and Vapnik, 1995) to classify sentiment.

2.2 Evaluating tool performance

For each of the tools studied in this work, we retrain the tool using the recommendations and procedures described in the paper that introduces the tool. We train each tool using the selected datasets using a stratified 70%/30% train-test split, as used by previous work (Novielli et al., 2020). To assess the performance of the sentiment-analysis tools for RQ_1, we study performance metrics like precision, recall, and f1. For RQ_2 we study both performance metrics *and* the inter-tool agreement on the test set. Both performance metrics and inter-tool agreement have been used previously to evaluate sentiment analysis tools (Novielli et al., 2020). To reduce the chances of a particular train/test split introducing a bias we take ten different train/test splits of each dataset and evaluate the performance of each tool on each split.

RQ_1: To answer this research questions we compare the observed performance scores of the best performing machine learning tool with the two transformer-based deep-learning tools. Because we run 10 train/test runs, we compare the obtained distributions of performance scores. This comparison

Table 1: Number of non-natural language elements identified in the 100 item sample of each dataset

Item	**GitHub**	**StackOverflow**
Code	17	4
Username	10	3
Url	4	4
Version Number	1	2
Filename / path	1	2
Warning / Error code	2	1
Command	1	-
Hash	1	-
Mail fragment	1	-
Total	38	16

is made per performance metric (f1, precision, recall) for the macro averaged scores over the three sentiment polarity classes.

Our null hypothesis for *RQ1* is the following:

– Hypothesis 1: *There is no difference in the predictive performance between deep-learning and machine-learning models for sentiment analysis in software engineering.*

To test this hypothesis we first apply a Kruskall-Wallis test to see if there is any difference between the performance scores. If the p-score is lower than 0.05, we apply a set of Dunn's tests as post-hoc tests: One per dataset and performance metric (Dinno, 2015). To correct for a false discovery rate we adjust p-values using the Benjami-Hochberg procedure (1995). We reject the hypothesis if the adjusted p-value is lower than 0.05, and confirm the alternative hypothesis that at least one model has different predictive performance.

RQ_2: To study whether meta-tokenization improves the ability of sentiment analysis tools to predict sentiment we first identify meta-tokens in the two datasets, and we study whether the usage of meta-tokens improves accuracy and agreement. The agreement of sentiment analysis tools has been studied previously in benchmarks (Novielli et al., 2020).

To identify meta-tokens we sampled 100 items from each dataset. This 200-item sample was manually labeled by two authors of the paper. The labeling task was to identify, extract, and name all non-natural language elements. For the labeling task, we define non-natural language elements as those elements that are not regular text, specifically, we consider class names that are used as named entities as natural language elements. After identifying and extracting the non-natural language elements each extracted element was labeled with a descriptive name by the labeler. The agreement between the two labelers was substantial, with a Cohen's kappa of 0.65. Any remaining conflicts, and the naming itself, were discussed in a shared session and any conflicts were resolved. The final extraction and naming of non-natural language elements are listed in Table 1.

Table 2: List of tokens, the expressions used to detect them, and the number of meta-tokens they are replaced with for both datasets.

Type	**Token**	# Replacements	
		Github	**StackOverflow**
Email	`M_EMAIL`	140	9
Username	`M_MENTION`	715	235
Inline Code	`M_ICODE`	89	126
Version Number	`M_VERSION_NUMBER`	342	172
Issue reference	`M_ISSUE_MENTION`	51	-
URL	`M_URL`	368	182

To replace the non-natural language elements in the dataset we use the following procedure: Based on the non-natural language elements identified (Table 1) we manually created a set of meta-tokens. This list is extended with non-natural language elements that occur in the markdown documentation of each platform. Each meta-token is a tuple of a regular expression and a token name. The tokens we use per dataset are listed Table 2, while the regex rules used to replace these tokens can be found in the replication package.[2] Each document in the dataset is then processed using these tuples, and each regular expression match is replaced with the token name. For example, if a code fragment is identified, we replace the code fragment with the meta-token `M_ICODE`. We maintain a separate list of meta-tokens per platform because the markup language used differs slightly per platform. The total number of replacements per meta-token is listed in Table 2. 22% of the items in the Github dataset contain at least one meta-token, and 13% of the items in the StackOverflow dataset contain at least one meta-token.

To measure the impact of meta-tokenization on predictive performance we take the median performance score of each tool for each dataset, for each performance metric, and for each sentiment polarity class. We compare the median score of that particular tool trained on the dataset, with the median score of that tool trained on the meta-tokenized version of the dataset. By comparing the median intra-tool scores we hope to understand whether meta-tokenization has an impact.

Moreover, we also use a Mann-Whitney Wilcoxon test (McKnight and Najab, 2010) to compare intra-tool performance scores for the tools trained on the meta-tokenized and untokenized versions of the dataset. We use a Mann-Whitney Wilcoxon test as opposed to Kruskall-Wallis with as post-hoc a Dunn's test since we are comparing the two distributions of performance scores for each tool. We adjust p-values using the Benjami-Hochberg procedure (1995) to adjust for a false discovery rate.

[2] The replication package can be found on figshare (`https://figshare.com/s/1dbdf605abb20441b3d8`), and for each platform, a notebook with the replacement rules exists.

To further understand the effects of meta-tokenization we compute the weighted Cohen's kappa (Cohen, 1968) per tool pair per run. We then use a similar statistical methodology for the predictive performance to study whether there is a statistical difference between inter-tool agreement for tools trained on the original version of the dataset and the meta-tokenized version of the dataset.

For the statistical tests, we use the following null hypotheses:

– Hypothesis 2: *There is no difference in the predictive performance between a sentiment analysis tool trained on a meta-tokenized version of a dataset vs. the same tool trained on an unmodified version of the dataset.*
– Hypothesis 3: *There is no difference in the intra-tool agreement between sentiment analysis tools trained on the meta-tokenized version of a dataset vs an unmodified version of the dataset.*

For Hypothesis 2 we test the hypothesis for each dataset, tool, and performance metric and adjust the obtained p-values accordingly. For Hypothesis 3 we test the hypothesis per tool pair and per dataset and adjust the p-values over these comparisons. We reject each hypothesis if the adjusted p-value is lower than 0.05. If Hypothesis 2 is rejected, we confirm the alternative hypothesis that there are differences in the predictive performance of the tool depending on the version of the dataset it is trained on. In the case that Hypothesis 3 is rejected, we confirm the alternative hypothesis that there are differences in the intra-tool agreement depending on the version of the dataset they are trained on.

3 Results

This section reports the performance of the machine-learning and deep-learning sentiment analysis tools (*RQ*$_1$). The section also reports the performance of sentiment analysis tools after retraining them on meta-tokenized versions of the datasets (*RQ*$_2$). **Data availability statement:** The dataset of performance scores of the analyzed sentiment-analysis tools is publicly available in a Figshare repository.[3]

3.1 Machine learning and Deep learning

Table 3 contains the results of the ten runs for each tool on each dataset. Boldface highlights the best-performing tool per metric. As can be observed, the two deep-learners outperform the machine-learning tool on the StackOverflow dataset. However, for the GitHub dataset there are instances where Senti4SD outperforms the deep-learners. When performance differences exist between the best performing machine-learner and the deep-learning tools these differences are mostly a few percentage points.

[3] `https://figshare.com/s/1dbdf605abb20441b3d8`

Table 3: Median performance score for each metric per tool for each of the ten runs.

		Positive			Negative			Neutral			Macro		
	Tools	*P*	*R*	*F1*	*P*	*R*	*F1*	*P*	*R*	*F1*	*P*	*R*	*F1*
GH	Senti4SD	0.937	0.906	0.919	**0.911**	0.887	**0.902**	0.891	0.924	**0.906**	**0.911**	0.906	**0.908**
	SentiSW	0.807	0.802	0.809	0.772	0.649	0.701	0.741	0.836	0.787	0.777	0.760	0.766
	SentiCR	0.893	0.842	0.869	0.867	0.695	0.774	0.780	0.915	0.842	0.848	0.820	0.829
	Sentimoji	**0.941**	0.919	**0.929**	0.907	0.845	0.876	0.872	**0.927**	0.899	0.907	0.898	0.902
	Bert	0.911	**0.950**	**0.929**	0.890	**0.891**	0.887	**0.927**	0.896	**0.906**	0.907	**0.912**	**0.908**
SO	Senti4SD	0.899	0.920	0.909	0.787	0.842	0.817	0.833	0.779	0.807	0.843	0.846	0.844
	SentiSW	0.866	0.886	0.882	0.820	0.712	0.763	0.780	0.836	0.806	0.822	0.812	0.815
	SentiCR	0.880	0.906	0.895	0.790	0.731	0.758	0.796	0.814	0.805	0.822	0.819	0.820
	Sentimoji	0.923	0.931	0.926	0.842	0.835	0.836	0.839	0.839	0.834	0.868	0.867	0.867
	Bert	**0.924**	**0.939**	**0.930**	**0.849**	**0.863**	**0.853**	**0.863**	**0.847**	**0.851**	**0.878**	**0.879**	**0.878**

Table 4: Table containing the results of the Dunn's tests for the comparison of deep-learners and machine-learners on the macro performance metrics.

Metric	**Tools**	**Corrected P-value**	
		GitHub	StackOverflow
f1	Senti4SD/Sentimoji	.031*	.010*
f1	Senti4SD/Bert	.629	< .001***
f1	Sentimoji/Bert	.108	.153
precision	Senti4SD/Sentimoji	.127	.010*
precision	Senti4SD/Bert	.127	< .001***
precision	Sentimoji/Bert	.959	.123
recall	Senti4SD/Sentimoji	.042*	.010*
recall	Senti4SD/Bert	.909	< .001***
recall	Sentimoji/Bert	.031*	.127

***: $p < 0.001$, **: $p < 0.01$, *: $p < 0.05$

To study the distribution of the performance scores, Figure 1 presents a violin plot of the three best-performing sentiment analysis tools (Senti4SD the best-performing machine learner, and Bert and Sentimoji two deep learners) on both datasets. The violin plot visualizes the macro-averaged performance scores per metric of the 10 runs per tool. As can be observed, the performance differences between most tools for most metrics on GitHub are small or hard to distinguish. Meanwhile, for the StackOverflow dataset, the performance differences between the three tools are easier to see.

The p-values for the Kruskall-Wallis tests are all smaller than .001. Therefore, we compare the performance scores obtained using Dunn's test. The P-values of these comparisons are shown in Table 4. For the GitHub dataset the only significant difference is found between Senti4SD and Sentimoji for recall and f1, and between Sentimoji and Bert for recall. Meanwhile, for the Stack-Overflow dataset we find that Bert and Sentimoji are different from Senti4SD for all performance metrics. Additionally, no statistically significant differences are found between Bert and Sentimoji.

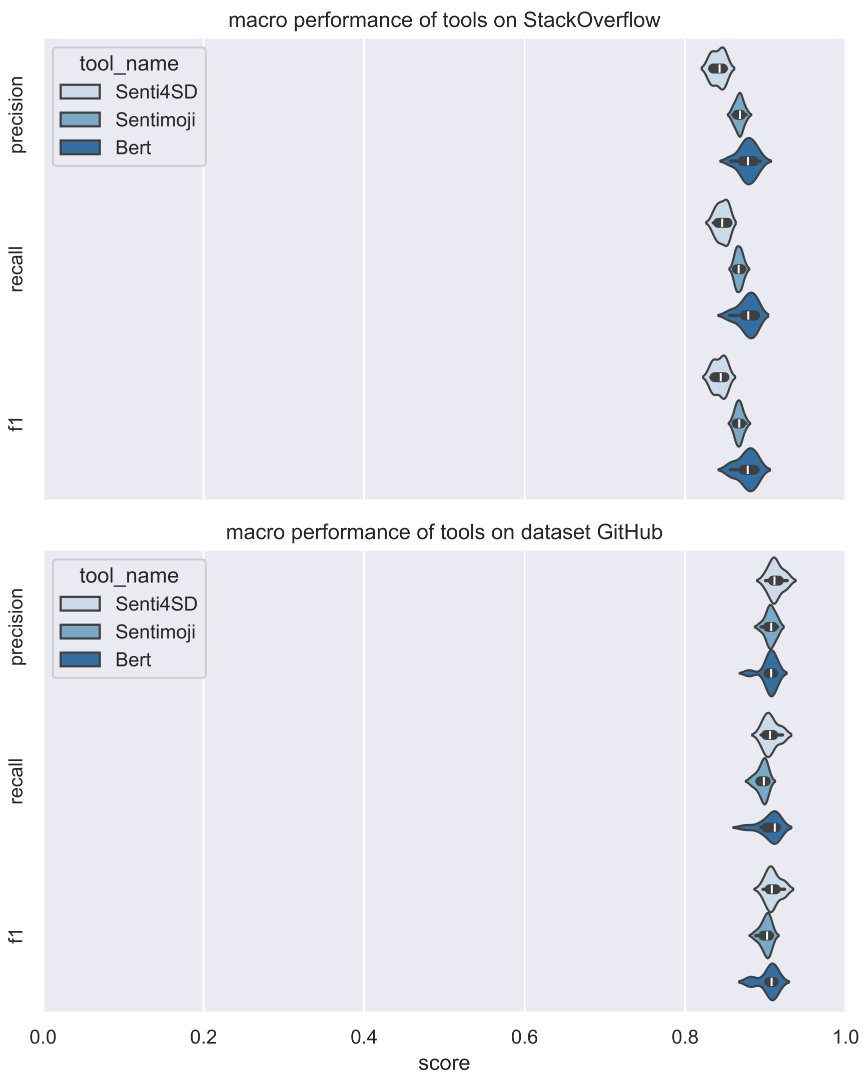

Fig. 1: Performance of Senti4SD, Sentimoji and BERT on the GitHub and StackOverflow datasets. The range on the y-axis for plots range from 0 to 1.

RQ_1

Transformer-based models outperform machine-learning tools on the StackOverflow dataset, while no significant performance differences are observed for the GitHub dataset. However, the observed performance differences between the best-performing machine-learner and transformer-based model are a few percentage points at most.

3.2 Meta-tokenization

Table 5 lists the median performance scores per sentiment polarity class and metric after benchmarking the five tools on the untokenized and meta-tokenized (mt) versions of the datasets. Each pair of rows corresponds to a tool and a dataset, and the boldface indicates on which version of the dataset the tool managed to score higher. In case of a tie, both values are typeset in bold.

As can be observed in Table 5 meta-tokenization does not appear to greatly affect the predictive performance of the three sentiment analysis tools. For some classes and for some tools the performance of the tools trained on the meta-tokenized version of the dataset appears to be slightly higher. However, the difference in performance scores for the tools trained on the untokenized and meta-tokenized datasets is quite small. SentiCR, for instance, scores slightly higher on most metrics and classes of the meta-tokenized version of the GitHub dataset than the untokenized version, however, these observed differences are minor.

To test whether any significant differences exist between the tools on meta-tokenized and untokenized versions of datasets, we use the Mann-Whitney U test to compare the distributions. However, we find that the adjusted p-values after running the pairwise Mann-Whitney U tests are all 1.0, for all tools on both datasets. Therefore, we observe no evidence that meta-tokenization influences predictive performance.

To determine whether meta-tokenization affects the agreement of the sentiment analysis tools we compute the weighted Cohen's Kappa for each tool pair per run and dataset. The corrected p-values for the pairwise Mann-Whitney U tests comparing the observed agreement before and after meta-tokenization are all 0.970 indicating that meta-tokenization does not affect the agreement of the tools.

RQ_2

We conclude, based on the performed benchmarks, that there is no evidence that meta-tokenization significantly improves either the predictive performance of sentiment analysis tools or the ability of sentiment analysis tools to agree.

4 Devil's Advocate

In this work we present negative results: Meta-tokenization does not improve predictive performance or agreement of sentiment analysis tools, and pre-trained transformers do not always outperform machine learning tools. Therefore, in this section, we present and answer several questions that could be raised by a *Devil's advocate* concerning the soundness of our methodology. Each subsection presents a question, a short motivation for the question, and

Table 5: Performance metric results of running the tools on the original datasets and tokenized versions of the datasets. Each row corresponds to the median performance score over the ten runs for that particular tool, "mt" is used to denote that the tool has been trained on the meta-tokenized version of the dataset.

Tools	Dataset	Positive			Negative			Neutral			Macro		
		Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
Senti4SD	GH	0.937	**0.906**	0.919	0.911	0.887	0.902	**0.891**	0.924	**0.906**	0.911	0.906	0.908
	GH (mt)	**0.943**	0.901	**0.924**	**0.913**	**0.888**	**0.905**	0.887	**0.926**	**0.906**	**0.915**	**0.908**	**0.911**
	SO	0.899	**0.920**	0.909	0.787	0.842	0.817	**0.833**	0.779	**0.807**	0.843	0.846	0.844
	SO (mt)	**0.901**	0.917	**0.912**	**0.797**	**0.843**	**0.820**	0.831	**0.781**	0.806	**0.844**	**0.848**	**0.846**
SentiSW	GH	**0.807**	**0.802**	**0.809**	**0.772**	0.649	0.701	0.741	**0.836**	0.787	**0.777**	0.760	0.766
	GH (mt)	**0.807**	0.800	0.800	0.765	**0.661**	**0.710**	**0.751**	0.830	**0.793**	**0.777**	**0.764**	**0.769**
	SO	**0.866**	**0.886**	**0.882**	**0.820**	0.712	**0.763**	0.780	0.836	0.806	**0.822**	**0.812**	**0.815**
	SO (mt)	0.865	0.883	0.881	0.817	**0.720**	0.761	**0.784**	**0.838**	**0.807**	0.820	0.811	0.814
SentiCR	GH	0.893	0.842	0.869	0.867	**0.695**	0.774	0.780	0.915	0.842	0.848	0.820	0.829
	GH (mt)	**0.897**	**0.851**	**0.874**	**0.878**	0.693	**0.777**	**0.783**	**0.920**	**0.848**	**0.853**	**0.824**	**0.833**
	SO	0.880	0.906	**0.895**	0.790	**0.731**	0.758	0.796	0.814	0.805	0.822	**0.819**	0.820
	SO (mt)	**0.881**	**0.909**	**0.895**	**0.795**	**0.731**	**0.763**	**0.799**	**0.823**	**0.809**	**0.826**	**0.819**	**0.821**
Sentimoji	GH	**0.941**	0.919	0.929	0.907	**0.845**	**0.876**	**0.872**	0.927	0.899	0.907	0.898	0.902
	GH (mt)	**0.941**	**0.925**	**0.935**	**0.912**	0.841	0.875	**0.872**	**0.928**	**0.902**	**0.909**	**0.900**	**0.904**
	SO	0.923	**0.931**	**0.926**	**0.842**	0.835	0.836	**0.839**	**0.839**	0.834	**0.868**	**0.867**	**0.867**
	SO (mt)	**0.925**	0.930	0.925	0.841	**0.837**	**0.838**	**0.839**	0.837	**0.835**	**0.868**	0.866	0.866
Bert	GH	0.911	**0.950**	**0.929**	0.890	**0.891**	**0.887**	**0.927**	0.896	0.906	0.907	**0.912**	0.908
	GH (mt)	**0.920**	0.940	0.926	**0.924**	0.857	**0.887**	0.901	**0.938**	**0.913**	**0.914**	**0.912**	**0.911**
	SO	**0.924**	0.939	0.930	0.849	0.863	0.853	0.863	**0.847**	0.851	0.878	0.879	0.878
	SO (mt)	0.901	**0.965**	**0.931**	**0.853**	**0.874**	**0.858**	**0.891**	0.834	**0.855**	**0.880**	**0.881**	**0.880**

a response. This section is inspired by the line of reasoning used by Sidhu et al. (2019).

4.1 What process was used to label the items in the dataset? Could bias in the labeling influence the results? Could bias in train-test splits influence the results?

From the work of Novielli et al. (2020) we know that labeling datasets used to train sentiment analysis tools is important, as datasets should be labeled using clear and consistent guidelines. Not only does the labeling of datasets matter, but how a dataset is split into a train and test split might also influence results.

The two datasets selected for this study, the StackOverflow and the GitHub datasets, have been labeled using labeling guidelines based on existing theories of affect. Additionally, for both datasets the labeling process was executed over several rounds, and for each round disagreements were discussed (Calefato et al., 2018a; Novielli et al., 2020). This ensures that for both datasets inter-rater reliability is high, resulting in robust and reliable datasets with a well-operationalized definition of sentiment. By training the tools on these gold-standard datasets we minimize the chances that ad-hoc labeling, or inconsistent labeling influences the performance of the tools.

Additionally, each experiment in this study is repeated ten times, each time using a different random seed for the stratified train-test split. Using this repetition we avoid that the results are influenced by a single train-test split or a single initialization of random parameters for one of the tools.To ensure that the tools are compared on equal grounds, the same train/test split is used across the tools for each run. By reporting both the median performance score, and by doing statistical testing on the obtained performance scores we aim to obtain results that are sound and reliable. These 10 runs ensure that for both research questions (RQ_1 and RQ_2) the obtained differences across tools, or as a result of meta-tokenization, are not due to random effects, or opportune train-test splits.

Response: Given the reliable labelling process on both datasets and our multiple runs with random train-test splits, our confidence on the RQ1 and RQ2 results is strengthened.

4.2 Don't sentiment analysis tools already apply preprocessing techniques to handle non-natural language?

If the existing sentiment analysis tools evaluated in this work already apply techniques to filter out or otherwise preprocess non-natural language the findings for RQ_2 *might be impacted.*

To determine whether the tools benchmarked in this study apply techniques that might affect the effectiveness of meta-tokenization we analyze the papers in which the tools were originally described (Ahmed et al., 2017; Biswas et al., 2020; Calefato et al., 2018a; Chen et al., 2019; Ding et al., 2018). From the papers we extract and read the sections in which the preprocessing process is described, and we report the steps taken by the tools to process the input texts. Where needed we also analyzed the available source-code of the tools, to better understand how the tools pre-process input.

Senti4SD: Uses extensive feature engineering which can be divided into three categories: generic sentiment lexicon features, keyword-based features and features based on word embeddings (Calefato et al., 2018a). While computing these features Senti4SD applies very limited preprocessing. It only replaces all usernames with the meta-token `@USERNAME`. However, Senti4SD does not perform any stemming or lemmatization, nor are stopwords removed. In the paper for Senti4SD no details are mentioned about the removal of URLs, stopwords or HTML. However, the authors do mention that the dataset on which Senti4SD was originally trained is a dataset in which URLs, code snippets and HTML tags were removed. This dataset is the StackOverflow gold-standard dataset.

SentiCR: Uses many different preprocessing steps to process an input item of text, in total 7 steps are used (Ahmed et al., 2017). In order of execution these are:

1. *Expansion of contraction*: Expands contractions such as *I'm* $\rightarrow$ *I am* using a dictionary of 124 commonly occurring contractions.
2. *URL Removal*: Removes all URLs from the text.
3. *Handling of emoticons*: Replaces four emoticons with a predefined token indicating whether the emoticon is positive or negative.
4. *Negation pre-processing*: Uses NLTK (Bird et al., 2009) to express a chunk grammar that can recognize and annotate negations such as *"I do not like your changes"*.
5. *Word stemming*: The stemming of input words with the stemmer of NLTK.
6. *Stop-word removal*: Removal of stop-words using a customized list of stop-words.
7. *Code-snippet removal*: The removal of code snippets through a list of predefined keywords, and the removal of all words that are present in less than three input texts of the train set.

SentiSW: Similar to SentiCR, SentiSW uses a preprocessing pipeline that contains the following steps in order of execution:

1. *Non-English character deletion*: The deletion of all non-ascii characters from the input.
2. *Contraction expansion*: Similar to SentiCR, however, no mention is made of the list of contraction used.
3. *Code snippet removal*: The usage of a GitHub markdown parser to remove markdown code snippets.
4. *URL and quotation removal*: Removal of URLs and text enclosed in quotes.
5. *Stop-word removal*: Removal of stop-words using a predefined list provided by StanfordNLP (Manning et al., 2014).

6. *Emoticons and punctuation mark processing*: The replacement of emoticons with tokens indicating whether the emoticon is positive or negative.
7. *Negation marking*: The usage of grammar rules to annotate negations, a predefined list of negation words is used.
8. *Word tokenization and stemming*: The usage of NLTK to tokenize and stem words (Bird et al., 2009).

SEntiMoji: In the paper of SEntiMoji no explicit mention of preprocessing of input data is made (Chen et al., 2019). The one detail that is mentioned is that the dataset used for the finetuning of SEntiMoji has been processed using meta-tokens for code, urls and issue references. However, no mention is made of applying this same preprocessing to input or training data. Through a manual investigation of the source-code of SEntiMoji we find that some sort of meta-tokenization is applied on input data. Namely, SEntiMoji replaces URLs, mentions using @ and URls in input data with meta-tokens.

BERT-based transformers: The paper itself makes no explicit mention of preprocessing that is applied (Biswas et al., 2020). However, in the source-code of the accompanying replication package we find that the authors use an existing tokenizer from the huggingface library[4]. This tokenizer is a Sentence-Piece tokenizer, which splits a sentence up into several smaller tokens (Kudo and Richardson, 2018). However, this tokenizer does not apply any meta-tokenization.

All studied sentiment analysis tools have different preprocessing pipelines. Conceptually, SentiCR and SentiSW are most similar, as they both use similar preprocessing pipelines, with differences in the implementation of certain steps. SEntiMoji falls between SentiCR and SentiSW as it does apply some form of meta-tokenization, however, it only applies this meta-tokenization for a limited number of tokens, as opposed to the meta-tokens identified in this work. Meanwhile Senti4SD has a very limited preprocessing timeline, and the BERT-based transformers both have a preprocessing pipeline that does not remove non-natural language.

Response: If the existing preprocessing applied by the tools influences the effectiveness of meta-tokenization one would expect to see that meta-tokenization is effective for Senti4SD and the BERT-based transformers, but not for SentiCR, SentiSW and SEntiMoji. However, we find no evidence for the effectiveness of meta-tokenization for any of the tools. Which allows us to conclude that the preprocessing steps already executed by the tools is not comparable to the meta-tokenization performed in this work.

5 Discussion

Through the experiments conducted in this work, we observe two different findings: We find limited evidence supporting the recommendation that large-scale deep-learning sentiment-analysis tools outperform existing machine-learning

[4] https://huggingface.co/docs/tokenizers/index

tools. Additionally, we find no evidence that meta-tokenization improves the performance of sentiment analysis tools.

5.1 Applying Sentiment Analysis Tools to Study Software Engineering

Improper usage of sentiment analysis tools might impact the replicability of studies that use sentiment-analysis tools to derive conclusions (Lin et al., 2022). Therefore, existing literature has studied how to apply sentiment analysis to software engineering data. As a result, there are many different recommendations on how to select and apply sentiment analysis tools. In benchmarks of general-purpose sentiment analysis tools applied to software engineering data Jongeling et al. (2017) found that general-purpose tools are not accurate. As a result, Jongeling *et al.* recommend using tools that are designed for software engineering data and tools that are tailored to the lingo used by developers. Novielli et al. (2020) recommend training sentiment analysis tools on gold-standard datasets. If no gold-standard dataset is available for a given context, Novielli et al. (2020) recommend using rule-based sentiment analysis tools. Additionally, Novielli *et al.* recommends that sentiment analysis tools should not be used outside of the platform. For instance, a tool trained on GitHub data should not be used to predict sentiment on StackOverflow data. Based on additional benchmarks Uddin et al. (2022) recommends using a supervised tool that combines the output of five state-of-the-art sentiment tools to achieve a 4% increase in accuracy over the best-performing standalone tool.

In addition to the recommendations on how to select sentiment analysis tools, there are also recommendations on how to analyze sentiment in software engineering: Novielli et al. (2020, 2021) recommend that sentiment analysis tools should always be validated on a robustly labeled sample of the data to ensure the tool is accurate. This process of labeling a small sample and validating the tool should continue until the tool is sufficiently accurate. Additionally, Novielli et al. (2020) recommend explicitly picking an established theory of affect and purposefully using sentiment analysis tools that align with this theory of affect. Finally, Novielli et al. (2021) recommend carefully considering the unit of analysis (sentences vs. documents).

In this paper, we add a more fine-grained recommendation on how to select sentiment analysis tools to the body of literature based on our results for RQ_1, namely: Given the relatively minor performance differences between the tools based on machine learning vs. deep learning, which we include in our benchmark (Table 5), *we recommend that the tool choice for sentiment analysis should not solely be based on predictive performance. Instead, the tool choice should depend on the alignment of the tool with the chosen theory of affect, domain adaption, and the suitability of the tool for the given task.* In practice, the choice of models bigger in terms of language model or tool complexity might not automatically result in a better performance. Instead, many other aspects are more important to ensure the validity of obtained results.

Table 6: The percentage of items per dataset, per polarity class that contain non-natural language that has been replaced with meta-tokens.

Dataset	**% Meta-tokens**		
	Negative	*Neutral*	*Positive*
GitHub	12.89%	35.24%	11.38%
StackOverflow	11.65%	15.05%	11.39%

The two studied datasets contain items that were sampled from two different platforms. This choice is in line with the intention to minimize the risk of platform or context influencing our results we have opted to use datasets from two different platforms. Specifically, two platforms were considered: Github, a collaborative software development platform, and StackOverflow, a Q&A platform, are used by software engineers to communicate with each other. However, the language used on these two platforms might differ from the language used on other platforms. Specifically, it might not be representative of language used in other software engineering contexts such as the language used during closed-source development. While most of the publicly available datasets of developer communication prepared for sentiment analysis have been derived from open-source projects or StackOverflow (Lin et al., 2022), there have been attempts to apply sentiment analysis to contexts, such as transcripts of meetings (Herrmann and Klunder, 2021). However, in their study the language of the meetings is German, and the datasets are not publicly available, making it infeasible to include data such as this in our study.

Suppose such datasets were available, it is not unthinkable that differences in language usage or communication norms might influence the performance of sentiment analysis tools. In other contexts, such as the study of Self-Admitted Technical Debt, it has been found that practices between open-source and industry differ (Zampetti et al., 2021). Previous studies show that sentiment analysis tools are sensitive to the dataset and the context or platform on which they have been trained (Novielli et al., 2020, 2021). Therefore, when transferring sentiment analysis tools to other contexts one should be aware of the potential limitations and the need to validate and potentially retrain sentiment analysis tools on the specific context.

5.2 Dataset creation and presence of non-natural language

For RQ_2 we studied the effect of meta-tokenization on two datasets, GitHub (Novielli et al., 2020) and StackOverflow (Calefato et al., 2018a). During the creation of the datasets the authors of both datasets made different decisions: According to the paper, Calefato et al. (2018a) removed code fragments, URLs and HTML from the text in the dataset. However, in the manual labeling and automatic removal of non-natural language elements (Tables 1 & 2) we still identified code fragments and URLs in the StackOverflow dataset. In the work of Calefato *et al.* they used a different approach to remove such elements than the Regex-based approach used in this work. Specifically, they only removed multi-line code elements using HTML parsing. In the GitHub dataset (Novielli et al., 2020), no mention is made of removing any non-natural language elements. This difference in approaches has replaced a greater number of source-code fragments with meta-tokens in the GitHub dataset than in the StackOverflow dataset.

Table 6 shows how many items were replaced with meta-tokens per sentiment polarity class in each dataset. Even though the process with which both datasets were created was different, the proportion of items with negative or positive sentiment that contain meta-tokens is similar for both datasets. However, there are more items with meta-tokens in the GitHub dataset for the neutral class than in the StackOverflow dataset. This difference in the proportion of meta-tokens in the GitHub dataset is why we intuitively expected meta-tokenization to work: Without meta-tokenization, sentiment analysis tools might learn to associate words used in non-natural language snippets with neutral sentiment. However, even on the GitHub dataset, no effect from meta-tokenization is observed for any of the benchmarked tools. For the StackOverflow dataset we also do not observe any impact of meta-tokenization. However, while creating the dataset Calefato et al. (2018a) removed some non-natural language elements. These removals may have impacted the distribution of non-natural language elements over the sentiment polarity classes and the results obtained. Nonetheless, because there is still some imbalance in the StackOverflow dataset (cf. Table 6) and we observed no difference in the GitHub dataset, we do not expect this to have influenced our results. In practice, this means that replacing non-natural language elements does not further improve the performance of sentiment analysis tools. Therefore, we recommend not replacing non-natural language elements with meta-tokens for sentiment analysis tasks.

However, the notion of using meta-tokens, or semantic categories, might be beneficial for other contexts in which sentiment analysis or opinion mining is applied. For instance, for the task of summarizing opinions expressed about APIs, a topic previously studied by Uddin and Khomh (2017). The idea of using a separate pre-processing step to merge semantically similar words (*performance, maintainability, usability*) into one token (*non-functionals*) could further improve the accuracy of summarization tools.

5.3 Benchmarking sentiment analysis tools

When sentiment analysis tools are benchmarked, the experimental set-up should attempt to adhere to existing recommendations where possible. In the benchmarks performed for RQ_1 our results differ from the results reported by Biswas et al. (2020) and Chen et al. (2019). However, both Biswas *et al.* and Chen *et al.* did not adhere to the recommendation of Novielli et al. (2020) to retrain all benchmarked tools on the datasets used in the study. As a result of this, both studies find larger differences in predictive performance between the machine-learners and deep-learners.

Additionally, for the experiments in this work, we ran each tool ten times with different train-test splits per experiment to ensure that the results do not depend on one particular train-test split. While analyzing the performance scores of the sentiment-analysis tools we noticed that for some tools, there is a large amount of variance in the performance scores (Figure 1). Therefore, when benchmarking sentiment analysis tools, and especially when reporting the results of a single run in which small performance differences are observed, one should be mindful of this variance. Techniques to address such inconsistencies in results such as repeating runs or k-fold cross validation (Hastie et al., 2009), already exist. However, our findings again stress that these techniques remain important for the study of sentiment analysis tools in software engineering.

6 Threats to Validity

In this section we describe the threats to internal, external and conclusion validity (Runeson and Höst, 2009).

6.1 Internal Validity

A potential threat to internal validity is the presence of undetected and unreplaced non-natural language elements in the datasets. These unreplaced non-natural language elements could affect the validity of our results, as these elements might impact the ability of the sentiment analysis tools to learn to classify sentiment. To mitigate this risk of this happening we labeled a sample of 100 items from each dataset and we identified the non-natural language elements present in the sample, such that the most frequent types of non-natural language have been identified. While labeling the sample for non-natural language elements a substantial agreement was obtained by the two authors who performed this labeling task.

The position of non-natural language in the text matters. For instance, while labeling we encountered frequent examples of items such as usernames, filenames, and source-code that were used more like named entities: "*Thank you* `@Username`" vs. "`@USERNAME`*. Thanks that was extremely helpful*". In the second case, the non-natural language element exists separately from the comment, while in the first case it is part of the comment. We opted to not label the first occurrence as non-natural language since one could interpret the occurrence of non-natural language as a part of the text. However, for the automatic detection and replacement of non-natural language elements with meta-tokens we used regular expressions, which were not able to distinguish between these two cases. This imprecise removal might further explain why we do not observe any effect of meta-tokenization.

Another risk that might have affected the obtained conclusions is our choice for regular expression to replace the identified non-natural language elements. By design, this approach is only able to detect the non-natural language elements that have been properly escaped with the markdown language of the platform from which the dataset was taken. However, during labeling we found instances of non-natural language elements which were not (properly) escaped with markdown. Because of the choice for regular expressions these instances were not replaced by meta-tokens and might have influenced the results and the observed impact of meta-tokenization.

The validity of the results for the benchmarking depend on the quality of the original datasets according to Novielli et al. (2020). Both datasets used in this study have been labeled using well-defined labeling guidelines. However, both datasets were created by researchers from the same research group, the items in both datasets were sampled using semi-random sampling techniques, and both datasets are based on older datadumps. While these factors might have affected the ability of the tools to 'learn' how to classify sentiment we have no reason to believe that any bias introduced by the construction of the datasets is specific to one type of tools.

6.2 External Validity

For this work, we used two gold-standard datasets to evaluate the effect of meta-tokenization. While these two datasets are the only two gold-standard datasets available of this size labeled using theoretical models of affect other datasets have been labeled in a more ad-hoc manner (Lin et al., 2022). Our results might not generalize over these datasets. However, given the ad-hoc labeling used for these datasets any difference in observed results (either for predictive performance or for the impact of meta-tokenization) might not be due to the nature of the datasets, but due to the ad-hoc labeling. Therefore, we have not opted to include these datasets in the study.

6.3 Conclusion Validity

We run multiple statistical tests to compare both the predictive performance and agreement of the sentiment analysis tools. However, because we ran statistical tests for each performance metric and each setting, we corrected the p-values to reduce the false discovery rate. The procedure we used for this is the Benjamini-Hochberg procedure (1995).

7 Related Work

Benchmarking studies of Sentiment Analysis tools for Software Engineering: Several studies have sought to benchmark the performance of sentiment analysis tools used for software engineering. Jongeling *et al.* found that general-purpose sentiment analysis tools are inaccurate when they are applied to technical texts (Jongeling et al., 2017). To address this concern several software engineering-specific sentiment analysis tools have been designed and benchmarked (Bosu et al., 2015; Calefato et al., 2018a; Chen et al., 2019; Ding et al., 2018; Islam and Zibran, 2018; Zhang et al., 2020). A benchmark of sentiment analysis tools performed by Novielli *et al.* found that the dataset on which a software engineering specific sentiment analysis tool is trained on greatly influences the performance of the tools (Novielli et al., 2020). Moreover, Novielli *et al.* found that the dataset on which a sentiment analysis tool has been trained not only influences predictive performance but also conclusions that can be obtained when applying sentiment analysis tools (Novielli et al., 2021). While these benchmarks evaluate the performance of sentiment analysis tools, they do not specifically investigate how meta-tokenization influences performance.

Some of the benchmarks performed to compare software-engineering sentiment analysis tools include a comparison of machine-learning tools and deep-learning tools(Chen et al., 2019; Zhang et al., 2020). In the work of Chen et al. (2019) the authors compare the performance of Sentimoji with several other sentiment-analysis tools, however, since the publication of the work newer datasets have been released. Therefore, in this benchmark study, we add a comparison between machine learning and deep learning tools on the GitHub gold-standard dataset. Moreover, in this work, we also compare Sentimoji with another deep learner: The BERT-based transformer. Meanwhile, the work of Zhang et al. (2020) compares BERT-based transformers with machine learning and dictionary-based sentiment analysis tools on both gold-standard datasets used for this work. However, in the work of Zhang *et al.* the authors do not retrain all machine-learning-based sentiment analysis tools in their benchmarks. In this study, we retrain all tools used for the study, both machine-learning and deep-learning-based ones, and therefore provide an accurate comparison. Uddin et al. (2022) also benchmark BERT-based transformers. In their work Uddin *et al.* compare the BERT-based transformer with an ensemble tool that combines the output of several sentiment analysis tools. They find that the ensemble tool (SentiSEAD) slightly outperforms the BERT-based transformers. However, they do not directly compare machine-learning-based sentiment analysis tools with the BERT-based transformers, and the work of Uddin *et al.* does not list the predictive performance of each of the tools used as part of the ensemble. Therefore, the paper does not contain enough information to answer RQ_1 .

Non-natural language in technical text: Previous work already studied the presence of non-natural language elements such as code fragments in technical texts. Bacchelli *et al.* investigated the usage of an automated technique to remove noise (code fragments) from e-mails (Bacchelli et al., 2010). They created a manually labeled dataset based on the mailing lists of several open-source projects and then used several features to classify whether a line belonging to an e-mail on the mailing list is natural language or not. While Bacchelli *et al.* study e-mail messages we use datasets that were taken from GitHub and StackOverflow, additionally, Bacchelli *et al.* perform a line-based classification while we replace tokens in sentences. Secondly, Bacchelli *et al.* do not study how their classification impacts sentiment analysis tools.

Mäntylä *et al.* designed an R-based classifier to classify whether a text fragment is natural-language or not (Mäntylä et al., 2018). Gathered data from several different platforms, and manually labeled whether these items are natural language. To classify whether a line of text is natural language or not Mäntylä *et al.* use a generalized linear model with a penalty, achieving high AUC and F-scores. However, in the work of Mäntylä *et al.* entire lines are classified, as opposed to the replacement of fragments within a text. Additionally, Mäntylä *et al.* do not study how this classification influences sentiment analysis tools.

Efstathiou *et al.* studied the language used by software engineers in code reviews, in their work they describe replacing non-natural language fragments with a token capturing the semantic meaning of the fragment (Efstathiou and Spinellis, 2018). While we apply an approach that is similar to that of Efstathiou *et al.* we study how these replacements influence the ability of sentiment analysis tools to classify sentiment.

8 Conclusion

In this work, we set out to answer two different research questions: Based on recent work (Fu and Menzies, 2017; Pamungkas et al., 2020; Shwartz-Ziv and Armon, 2022; Yedida and Menzies, 2022) we posed *RQ1*: *Do existing deep-learning sentiment analysis models outperform machine-learning-based sentiment analysis tools?* Secondly, based on the idea proposed by Efstathiou and Spinellis (2018) we posed *RQ2*: *How does the replacement of non-natural language elements in sentiment analysis data with meta-tokens affect the performance of Sentiment Analysis tools?*

We have taken five sentiment analysis tools designed for software engineering to answer these two research questions. Three machine-learning tools and two deep-learning-based tools. Additionally, we selected two gold-standard datasets of sentiment polarity and benchmarked all five tools on both datasets. To answer *RQ1*, we compared machine-learning tools' performance scores with the deep-learning-based tools' scores. To address *RQ2*, we identified and replaced several types of non-natural language elements in the dataset with meta-tokens. We then compared per tool an instance of the tool trained on the original dataset, and an instance of the tool trained on the meta-tokenized version of the dataset.

For *RQ1* we only observe minimal performance differences (no more than 4 percentage points) between the best-performing machine-learning tool (Senti4SD) and the two deep-learning-based sentiment analysis tools. Based on these findings, we extend the existing recommendations in the field of sentiment analysis for software engineering with the recommendation that the tool selection should not just be based on predictive performance. Instead, the alignment of the tool with the chosen theory of affect, and the tool's suitability for the given task should be considered. This recommendation holds as long as the chosen tools are trained with appropriate gold-standard datasets and the performance of these tools is validated on a robustly labeled sample. While deep-learning and machine-learning-based tools perform similarly when gold-standard datasets are available, future work could focus on understanding whether tools perform equally well in cases where less robustly labeled data is available.

Moreover, after studying the impact of meta-tokenization on the accuracy of sentiment analysis tools (RQ_2) we conclude that meta-tokenization does not improve predictive performance, or agreement. Based on this finding, we argue that the non-natural language elements present in the current gold-standard datasets does not reduce the ability of sentiment analysis tools to predict sentiment. However, there might be other contexts or domains in which non-natural language elements impact sentiment analysis tools' ability to predict sentiment, such as templated messages used by software bots. Future work could focus on understanding whether meta-tokenization is beneficial in these contexts.

Declarations

Conflict of Interests

The authors declared that they have no conflict of interest.

References

Ahmed T, Bosu A, Iqbal A, Rahimi S (2017) SentiCR: A customized sentiment analysis tool for code review interactions. ASE 2017 - Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering pp 106–111, DOI 10.1109/ASE.2017.8115623

Bacchelli A, D'Ambros M, Lanza M (2010) Extracting source code from e-mails. In: Proceedings of the 2010 IEEE 18th International Conference on Program Comprehension, IEEE Computer Society, USA, ICPC '10, p 24–33, DOI 10.1109/ICPC.2010.47, URL `https://doi.org/10.1109/ICPC.2010.47`

Benjamini Y, Hochberg Y (1995) Controlling the false discovery rate: A practical and powerful approach to multiple testing. Journal of the Royal Statistical Society Series B (Methodological) 57(1):289–300, DOI http://dx.doi.org/10.2307/2346101, URL `http://dx.doi.org/10.2307/2346101`

Bird S, Klein E, Loper E (2009) Natural language processing with Python: analyzing text with the natural language toolkit. " O'Reilly Media, Inc."

Biswas E, Karabulut ME, Pollock L, Vijay-Shanker K (2020) Achieving Reliable Sentiment Analysis in the Software Engineering Domain using BERT. Proceedings - 2020 IEEE International Conference on Software Maintenance and Evolution, ICSME 2020 pp 162–173, DOI 10.1109/ICSME46990.2020.00025

Bosu A, Greiler M, Bird C (2015) Characteristics of useful code reviews: An empirical study at microsoft. In: Proceedings of the International Conference on Mining Software Repositories, URL `https://www.microsoft.com/en-us/research/publication/characteristics-of-useful-code-reviews-an-empirical-study-at-microsoft/`

Calefato F, Lanubile F, Maiorano F, Novielli N (2018a) Sentiment Polarity Detection for Software Development. Empirical Software Engineering 23(3):1352–1382, DOI 10.1007/s10664-017-9546-9

Calefato F, Lanubile F, Novielli N (2018b) How to ask for technical help? Evidence-based guidelines for writing questions on Stack Overflow. Information and Software Technology 94(September 2017):186–207, DOI 10.1016/j.infsof.2017.10.009, URL `http://dx.doi.org/10.1016/j.infsof.2017.10.009`

Chen Z, Cao Y, Lu X, Mei Q, Liu X (2019) Sentimoji: An emoji-powered learning approach for sentiment analysis in software engineering. In: Proceedings of the 27th edition of ESEC/FSE, Association for Computing Machinery, ESEC/FSE 2019, p 841–852, DOI 10.1145/3338906.3338977, URL `https://doi.org/10.1145/3338906.3338977`

Cohen J (1968) Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. DOI 10.1037/h0026256

Cortes C, Vapnik V (1995) Support-vector networks. Machine learning 20(3):273–297

Ding J, Sun H, Wang X, Liu X (2018) Entity-level sentiment analysis of issue comments. Proceedings - International Conference on Software Engineering pp 7–13, DOI 10.1145/3194932.3194935

Dinno A (2015) Nonparametric pairwise multiple comparisons in independent groups using dunn's test. The Stata Journal: Promoting communications on statistics and Stata 15:292–300, DOI 10.1177/1536867X1501500117, URL `http://journals.sagepub.com/doi/10.1177/1536867X1501500117`

Efstathiou V, Spinellis D (2018) Code review comments: Language matters. In: ICSE NIER, ACM, p 69–72

Fu W, Menzies T (2017) Easy over hard: a case study on deep learning. ACM, pp 49–60, DOI 10.1145/3106237.3106256

Hastie T, Tibshirani R, Friedman J (2009) Model Assessment and Selection, Springer New York, New York, NY, pp 219–259. DOI 10.1007/978-0-387-84858-7_7, URL `https://doi.org/10.1007/978-0-387-84858-7_7`

Herrmann M, Klunder J (2021) From textual to verbal communication: Towards applying sentiment analysis to a software project meeting. IEEE, pp 371–376, DOI 10.1109/REW53955.2021.00065

Islam MR, Zibran MF (2018) Sentistrength-se: Exploiting domain specificity for improved sentiment analysis in software engineering text. Journal of Systems and Software 145:125–146, DOI https://doi.org/10.1016/j.jss.2018.08.030, URL `https://www.sciencedirect.com/science/article/`

pii/S0164121218301675

Jongeling R, Sarkar P, Datta S, Serebrenik A (2017) On negative results when using sentiment analysis tools for software engineering research. Empirical Software Engineering 22(5):2543–2584, DOI 10.1007/s10664-016-9493-x

Kudo T, Richardson J (2018) Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. Association for Computational Linguistics, pp 66–71, DOI 10.18653/v1/D18-2012

Lanovaz MJ, Adams B (2019) Comparing the communication tone and responses of users and developers in two r mailing lists: Measuring positive and negative emails. IEEE Software 36(5):46–50, DOI 10.1109/MS.2019.2922949

Lin B, Zampetti F, Bavota G, Penta MD, Lanza M, Oliveto R (2018) Sentiment analysis for software engineering. ACM, pp 94–104, DOI 10.1145/3180155.3180195, URL https://dl.acm.org/doi/10.1145/3180155.3180195

Lin B, Cassee N, Serebrenik A, Bavota G, Novielli N, Lanza M (2022) Opinion mining for software development: A systematic literature review. ACM Transactions on Software Engineering and Methodology 31:1–41, DOI 10.1145/3490388, URL https://dl.acm.org/doi/10.1145/3490388

Maalej W, Nabil H (2015) Bug report, feature request, or simply praise? on automatically classifying app reviews. In: 2015 IEEE 23rd International Requirements Engineering Conference (RE), pp 116–125, DOI 10.1109/RE.2015.7320414

Manning CD, Surdeanu M, Bauer J, Finkel JR, Bethard S, McClosky D (2014) The stanford corenlp natural language processing toolkit. In: ACL (System Demonstrations), The Association for Computer Linguistics, pp 55–60

Mäntylä M, Calefato F, Claes M (2018) Natural language or not (nlon) - a package for software engineering text analysis pipeline. In: 2018 IEEE/ACM 15th International Conference on Mining Software Repositories (MSR), p 387–391, DOI 10.1145/3196398.3196444

McKnight PE, Najab J (2010) Mann-Whitney U Test, John Wiley & Sons, Ltd, pp 1–1. DOI 10.1002/9780470479216.corpsy0524, URL https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470479216.corpsy0524

Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. In: Bengio Y, LeCun Y (eds) 1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings, URL http://arxiv.org/abs/1301.3781

Novielli N, Calefato F, Dongiovanni D, Girardi D, Lanubile F (2020) Can we use se-specific sentiment analysis tools in a cross-platform setting? In: Proceedings of the 17th International Conference on Mining Software Repositories, Association for Computing Machinery, New York, NY, USA, MSR '20, p 158–168, DOI 10.1145/3379597.3387446, URL https://doi.org/10.1145/3379597.3387446

Novielli N, Calefato F, Lanubile F, Serebrenik A (2021) Assessment of off-the-shelf se-specific sentiment analysis tools: An extended replication study. Empir Softw Eng 26(4):77, DOI 10.1007/s10664-021-09960-w, URL https:

//doi.org/10.1007/s10664-021-09960-w

Ortu M, Marchesi M, Tonelli R (2019) Empirical analysis of affect of merged issues on github. In: 2019 IEEE/ACM 4th International Workshop on Emotion Awareness in Software Engineering (SEmotion), pp 46–48, DOI 10.1109/SEmotion.2019.00017

Pamungkas EW, Basile V, Patti V (2020) Misogyny detection in twitter: a multilingual and cross-domain study. Information Processing & Management 57:102360, DOI 10.1016/j.ipm.2020.102360

Paul R, Bosu A, Sultana KZ (2019) Expressions of sentiments during code reviews: Male vs. female. In: 2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER), pp 26–37, DOI 10.1109/SANER.2019.8667987

Pennacchiotti M, Popescu AM (2011) Democrats, republicans and starbucks afficionados: User classification in twitter. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Association for Computing Machinery, New York, NY, USA, KDD '11, p 430–438, DOI 10.1145/2020408.2020477, URL https://doi.org/10.1145/2020408.2020477

Runeson P, Höst M (2009) Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering 14:131–164, DOI 10.1007/s10664-008-9102-8, URL http://portal.acm.org/citation.cfm?id=1519313.1519324

Shwartz-Ziv R, Armon A (2022) Tabular data: Deep learning is not all you need. Information Fusion 81:84–90, DOI 10.1016/j.inffus.2021.11.011

Sidhu PK, Mussbacher G, McIntosh S (2019) Reuse (or lack thereof) in travis ci specifications: An empirical study of ci phases and commands. IEEE, pp 524–533, DOI 10.1109/SANER.2019.8668029

Stanojevic M, Vraneš S (2009) Semantic approach to knowledge representation and processing. DOI 10.4018/978-1-60566-650-1.ch001

Uddin G, Khomh F (2017) Opiner: An opinion search and summarization engine for apis. IEEE, pp 978–983, DOI 10.1109/ASE.2017.8115715, URL http://ieeexplore.ieee.org/document/8115715/

Uddin G, Khomh F (2021) Automatic mining of opinions expressed about apis in stack overflow. IEEE Transactions on Software Engineering 47(3):522–559, DOI 10.1109/TSE.2019.2900245

Uddin G, Guéhénuc YG, Khomh F, Roy CK (2022) An empirical study of the effectiveness of an ensemble of stand-alone sentiment detection tools for software engineering datasets. ACM Transactions on Software Engineering and Methodology 31:1–38, DOI 10.1145/3491211

Yedida R, Menzies T (2022) How to improve deep learning for software analytics (a case study with code smell detection). pp 156–166

Zampetti F, Fucci G, Serebrenik A, Penta MD (2021) Self-admitted technical debt practices: a comparison between industry and open-source. Empirical Software Engineering 26:131, DOI 10.1007/s10664-021-10031-3, URL https://link.springer.com/10.1007/s10664-021-10031-3

Zhang T, Xu B, Thung F, Haryono SA, Lo D, Jiang L (2020) Sentiment analysis for software engineering: How far can pre-trained transformer models go? In: 2020 ICSME, pp 70–80, DOI 10.1109/ICSME46990.2020.00017