The Journal of Systems & Software 145 (2018) 125–146

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

SentiStrength-SE: Exploiting domain specificity for improved sentiment analysis in software engineering text

Md Rakibul Islam*, Minhaz F. Zibran

University of New Orleans, New Orleans, LA, United States

ARTICLE INFO

Keywords:
Sentiments
Emotions
Software engineering
Empirical study
Automation
Domain dictionary

ABSTRACT

Automated sentiment analysis in software engineering textual artifacts has long been suffering from inaccuracies in those few tools available for the purpose. We conduct an in-depth qualitative study to identify the difficulties responsible for such low accuracy. Majority of the exposed difficulties are then carefully addressed through building a domain dictionary and appropriate heuristics. These domain-specific techniques are then realized in `SentiStrength-SE`, a tool we have developed for improved sentiment analysis in text especially designed for application in the software engineering domain.

Using a benchmark dataset consisting of 5,600 manually annotated JIRA issue comments, we carry out both qualitative and quantitative evaluations of our tool. We also separately evaluate the contributions of individual major components (i.e., domain dictionary and heuristics) of `SentiStrength-SE`. The empirical evaluations confirm that the domain specificity exploited in our `SentiStrength-SE` enables it to substantially outperform the existing *domain-independent* tools/toolkits (`SentiStrength`, `NLTK`, and `Stanford NLP`) in detecting sentiments in software engineering text.

1. Introduction

Emotions are an inseparable part of human nature, which influence people's activities and interactions, and thus emotions affect task quality, productivity, creativity, group rapport and job satisfaction (Choudhury and Counts, 2013). Software development, being highly dependent on human efforts and interactions, is more susceptible to emotions of the practitioners. Hence, a good understanding of the developers' emotions and their influencing factors can be exploited for effective collaborations, task assignments (Dewan, 2015), and in devising measures to boost up job satisfaction, which, in turn, can result in increased productivity and projects' success.

Several studies have been performed in the past for understanding the role of human aspects on software development and engineering. Some of those earlier studies address *when* and *why* employees get affected by emotions (Choudhury and Counts, 2013; Guzman et al., 2014; Guzman and Bruegge, 2013; Pletea et al., 2014; Tourani et al., 2014), whereas some other work address *how* (Graziotin et al., 2013; Islam and Zibran, 2016a; 2016b; Lesiuk, 2005; Mäntylä et al., 2016; Murgia et al., 2014; Wrobel, 2013; 2016) the emotions impact the employees' performance at work.

Attempts are made to capture the developers' emotions in the workplace by means of traditional approaches such as, interviews, surveys (Wrobel, 2013), and biometric measurements (McDuff et al., 2012). Capturing emotions with the traditional approaches is more challenging for projects relying on geographically distributed team settings and voluntary contributions (e.g., open-source projects) (Guzman et al., 2014; Destefanis et al., 2015). Moreover, the traditional approaches involving direct observations and interactions with the developers often hinder their natural workflow. Thus, to supplement or complement those traditional approaches, recent attempts detect sentiments from the software engineering textual artifacts such as issue comments (Guzman et al., 2014; Pletea et al., 2014; Islam and Zibran, 2016a; 2016b; Mäntylä et al., 2016; Ortu et al., 2015; Calefato and Lanubile, 2016; Chowdhury and Hindle, 2016), email contents (Tourani et al., 2014; Garcia et al., 2013), and forum posts (Guzman and Bruegge, 2013; Novielli et al., 2014).

For automated extraction of sentiments from textual artifacts in the software engineering domain, three tools (i.e., `SentiStrength` (Thelwall et al., 2012), `NLTK` (Natural Language Toolkit) (NLTK, 0000), and `Stanford NLP` (Socher et al., 2013b)) are used while the use of `SentiStrength` is found dominant (Jongeling et al., 2015; Novielli, 0000). However, software engineering studies (Pletea et al., 2014; Tourani et al., 2014; Islam and Zibran, 2016a; Calefato and Lanubile, 2016; Chowdhury and Hindle, 2016;

* Corresponding author.
E-mail addresses: mislam3@uno.edu (M.R. Islam), zibran@cs.uno.edu (M.F. Zibran).

https://doi.org/10.1016/j.jss.2018.08.030
Received 3 July 2017; Received in revised form 18 June 2018; Accepted 10 August 2018
Available online 11 August 2018

Jongeling et al., 2015; Novielli et al., 2015; Tourani and Adams, 2016) involving sentiment analysis repeatedly report concerns about the accuracy of those sentiment analysis tools in the detection of sentimental polarities (i.e., negativity, positivity, and neutrality) of plain text contents. For example, when applied in the software engineering domain, SentiStrength and NLTK are respectively reported to have only 29.56% and 52.17% precision in identifying positive sentiments, and even lower precision of 13.18% and 23.45% respectively in the detection of negative sentiments (Tourani et al., 2014; Jongeling et al., 2015).

Those sentiment analysis tools are developed and trained using data from non-technical social networking media (e.g., twitter posts, forum posts, movie reviews) and when operated in a technical domain such as software engineering, their accuracy substantially degrades largely due to domain-specific variations in meanings of frequently used technical terms. Although such a domain dependency is indicated as a general difficulty against automated sentiment analysis in textual content, we need a deeper understanding of why and how such domain dependencies affect the performance of the tools, and how we can mitigate them. Indeed, the software engineering community demands a more accurate automatic sentiment analysis tool (Pletea et al., 2014; Tourani et al., 2014; Islam and Zibran, 2016b; Calefato and Lanubile, 2016; Chowdhury and Hindle, 2016; Novielli et al., 2015; Ortu et al., 2016b; Sinha et al., 2016). In this regard, this paper makes three major contributions:

- Using a large benchmark dataset, we carry out an in-depth exploratory study for exposing the difficulties in automatic sentiment analysis in textual content in a technical domain such as software engineering.
- We develop a *domain dictionary* specific for software engineering text. To the best of our knowledge, this is the first domain-specific sentiment analysis dictionary for the software engineering domain.
- We propose techniques and realize those in SentiStrength-SE, a prototype tool that we develop for improved sentiment analysis in software engineering textual content. The tool is also made freely available online (SentiStregth-SE, 0000). SentiStrength-SE is the first domain-specific sentiment analysis tool especially designed for software engineering text.

Instead of building a tool from scratch, we develop our SentiStrength-SE on top of SentiStrength (Thelwall et al., 2012), which, till date, is the most widely used tool for automated sentiment analysis in software engineering (Islam and Zibran, 2017b). From quantitative comparison with the original SentiStrength (Thelwall et al., 2012), NLTK and Stanford NLP as operated in the software engineering domain, we find that our domain-specific SentiStrength-SE significantly outperforms those domain independent tools/toolkits. We also separately evaluate the contributions of individual major components (i.e., the domain dictionary and heuristics) of our SentiStrength-SE in sentiment analysis in software engineering text. Our evaluations demonstrate that, for software engineering text, domain-specific sentiment analysis techniques perform substantially better in detecting sentiments accurately. We further conduct a qualitative evaluation of our tool. Based on the exploratory study and the qualitative evaluation, we outline plans for further improvements in automated sentiment analysis in the software engineering area.

This paper is a significant extension to our recent work (Islam and Zibran, 2017b). This paper presents *new evidence and insights* by including a *deeper analysis* of the difficulties in automated sentiment analysis in software engineering text. The techniques applied in the development of SentiStrength-SE are described in *greater detail*. The empirical evaluation of the tool is substantially extended with *deeper qualitative analyses and direct comparisons* with NLTK and Stanford NLP in addition to the previously published comparison with the original SentiStrength. We include separate evaluations of the individual major components (i.e., the domain dictionary and heuristics) of SentiStrength-SE. The quantitative comparisons are validated in the light of statistical tests of significance.

Outline: The rest of the paper is organized as follows. Section 2 describes a qualitative empirical study that reveals the challenges in automated sentiment analysis in software engineering. In Section 3, we introduce SentiStrength-SE, the prototype tool, that we have developed by addressing the identified difficulties. Section 4, presents quantitative and qualitative evaluation of our tool. In Section 4.9 we discuss the threats to validity of the empirical evaluation. In Section 5, we discuss scopes for further improvements and future research directions. Related work is discussed in Section 6. Finally, Section 7 concludes the paper.

2. Exploratory study of the difficulties in sentiment analysis

To explore the difficulties in automated sentiment detection in text, we conduct our qualitative analysis around the *Java version* of SentiStrength (Thelwall et al., 2012). This *Java version* is the latest release of SentiStrength, while the older version, strictly for use on Windows platform, is still available. As mentioned before, SentiStrength is a state-of-the-art sentiment analysis tool most widely adopted in the software engineering community. The reasons for choosing this particular tool are further justified in Section 6.

English dictionaries consider the words 'emotion' and 'sentiment' as synonymous, and accordingly the words are often used in practice. Although there is arguably a subtle difference between the two, in describing this work, we consider them synonymous. We formalize that, aside from subjectivity, a human expression can have two perceivable dimensions: sentimental *polarity* and sentimental *intensity*. Sentimental *polarity* indicates the positivity, negativity, or neutrality of expression while sentimental *intensity* captures the strength of the emotional/sentimental expression, which sentiment analysis tools often report in numeric emotional scores.

2.1. Benchmark data

In our work, we use a "Gold Standard" dataset (Ortu et al., 2016b; Gold Standard Dataset Labeled with Manually Annotated Emotions, 0000), which consists of 5,992 issue comments extracted from JIRA issue tracking system. The entire dataset is divided in three groups named as Group-1, Group-2 and Group-3 containing 392, 1,600 and 4,000 issue comments respectively. Each of the 5,992 issue comments are manually interpreted by n distinct human raters (Ortu et al., 2016b) and annotated with emotional expressions as found in those comments. For Group-1, $n = 4$ while for Group-2 and Group-3, $n = 3$. This is the only publicly available such dataset in the software engineering domain (Ortu et al., 2016b; Islam and Zibran, 2017b).

A closed set $\mathcal{E}$ of *emotional expressions* are used in the annotation of the issue comments in the dataset, where $\mathcal{E}$={*joy, love, surprise, anger, sad, fear*}. The human raters labeled each of the issue comments depending on whether or not they found the sentimental expressions in the comments. Formally,

$$\mathcal{F}^{r_j}_{\mathcal{E}_i}(C) = \begin{cases} 1, & \text{if emotion } \mathcal{E}_i \text{is found in } C \text{by rater } r_j. \\ 0, & \text{otherwise.} \end{cases}$$

An example of human annotations of an issue comment from the dataset is shown in Table 1.

2.2. Emotional expressions to sentimental polarities

Emotional expressions *joy* and *love* convey *positive* sentimental polarity, while *anger, sadness*, and *fear* express *negative* polarity. In some cases, an expression of *surprise* can be positive in polarity, denoted as

Table 1
Example of annotation of an issue comment by four human raters

Issue comment (Comment ID-53257): `Thanks for the patch; Michale. Applied with a few modifications.`

Human	Emotions ($\mathcal{E}_i$)					
Raters (r_j)	Joy	Love	Surprise	Anger	Sadness	Fear
Rater-1 (r_1)	**1**	**1**	0	0	0	0
Rater-2 (r_2)	0	0	0	0	0	0
Rater-3 (r_3)	**1**	0	0	0	0	0
Rater-4 (r_4)	**1**	0	0	0	0	0

Interpretation: rater-1 found 'joy' and 'love' in the comment, while rater-3 and rater-4 found the presence of only 'love' but rater-2 did not identify any of the emotional expressions.

surprise$^+$, while other cases can convey a *negative surprise*, denoted as *surprise*$^-$. Thus the issue comments in the benchmark dataset, which are annotated with *surprise* expression, need to be further distinguished based on the sentimental polarities they convey. Hence, we get each of such comments reinterpreted by three additional human (computer science graduate students) raters, who independently determine polarities of the surprise expressions in each comments.

We consider a *surprise* expression in a comment polarized negatively (or positively), if two of the three rates identify negative (or positive) polarity in it. We found 79 issue comments in the benchmark dataset, which were annotated with the *surprise* expression. 20 of them express *surprise* with positive polarity and the rest 59 convey negative *surprise*.

Then we split the set $\mathcal{E}$ of emotional expressions into two disjoint sets as $\mathcal{E}_+ = \{joy, love, surprise^+\}$ and $\mathcal{E}_- = \{anger, sad, fear, surprise^-\}$. Thus, $\mathcal{E}_+$ contains only the positive sentimental expressions and $\mathcal{E}_-$ contains only the negative sentimental expressions. A similar approach is also used in other studies (Jongeling et al., 2015; 2017) to categorize emotional expressions according to their polarities.

2.3. Computation of emotional scores from human rated dataset

For each of the issue comments in the "Gold Standard" dataset, we compute sentimental polarity using the polarity labels assessed by the human raters. For an issue comment C rated by n human raters, we compute a pair $\langle \rho_c^{r_j}, \eta_c^{r_j} \rangle$ of values for each of the n raters r_j (where $1 \le j \le n$) using Eq. 1 and Eq. 2:

$$\rho_c^{r_j} = \begin{cases} 1, & \text{if } \sum_{\mathcal{E}_i \in \mathcal{E}_+} \mathcal{F}_{\mathcal{E}_i}^{r_j}(C) > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

$$\eta_c^{r_j} = \begin{cases} 1, & \text{if } \sum_{\mathcal{E}_i \in \mathcal{E}_-} \mathcal{F}_{\mathcal{E}_i}^{r_j}(C) > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Thus, if a rater r_j finds the presence of any of the positive sentimental expressions in the comment C, then $\rho_c^{r_j} = 1$, otherwise $\rho_c^{r_j} = 0$. Similarly, if any of the negative sentimental expressions are found in the comment C, then $\eta_c^{r_j} = 1$, otherwise $\eta_c^{r_j} = 0$.

An issue comment C is considered neutral in sentimental polarity, if we get the pairs $\langle \rho_c^{r_j}, \eta_c^{r_j} \rangle$ for at least $n-1$ (i.e., majority) raters where $\rho_c^{r_j} = 0$ and $\eta_c^{r_j} = 0$. If the comment is not neutral, then we determine the positive and negative sentimental polarities of that issue comment. To do that, using the following equations, we count the number of human raters, $\mathcal{R}_+(C)$ who found positive sentiment in the comment C and also the number of raters, $\mathcal{R}_-(C)$, who found negative sentiment in the comment C.

$$\mathcal{R}_+(C) = \sum_{j=1}^{n} \rho_c^{r_j} \quad \text{and} \quad \mathcal{R}_-(C) = \sum_{j=1}^{n} \eta_c^{r_j} \tag{3}$$

An issue comment C is considered exhibiting positive sentiment, if at least $n-1$ human raters found positive sentiment in the message. Similarly, we consider a comment having negative sentiment if at least $n-1$ raters found negative sentiment in it. Finally, we compute the sentimental polarities of an issue comment C as a pair $\langle \rho_c^h, \eta_c^h \rangle$ using Eq. 4 and Eq. 5.

$$\rho_c^h = \begin{cases} 0, & \text{if } C \text{is neutral} \\ +1, & \text{if } \mathcal{R}_+(C) \ge n-1 \\ -1, & \text{otherwise.} \end{cases} \tag{4}$$

$$\eta_c^h = \begin{cases} 0, & \text{if } C \text{is neutral} \\ +1, & \text{if } \mathcal{R}_-(C) \ge n-1 \\ -1, & \text{otherwise.} \end{cases} \tag{5}$$

Thus, $\rho_c^h = 1$, only if the comment C has positive sentiment and $\eta_c^h = 1$ only if the comment contains negative sentiment. Note that, a given comment can exhibit both positive and negative sentiments at the same time. A comment is considered sentimentally neutral when the pair $\langle \rho_c^h, \eta_c^h \rangle$ for the comment appear to be $\langle 0, 0 \rangle$. An issue comment is discarded from our study if at least $n-1$ human raters (i.e., majority) could not agree on any particular sentimental polarity of the comment. We have found 33 such comments in Group-2 dataset that are excluded from our study. Similar approach is also followed to determine sentiments of comments in another study (Jongeling et al., 2015).

2.3.1. Illustrative example of computing sentimental polarity

Consider the issue comment in Table 1. For this issue comment, we compute the pair $\langle \rho_c^{r_j}, \eta_c^{r_j} \rangle$ for all four raters (i.e., $n = 4$). As for only one (the second rater) out of four raters we get the pair as $\langle 0, 0 \rangle$, the comment is not considered neutral. Hence, we compute the values of $\mathcal{R}_+(C)$ and $\mathcal{R}_-(C)$, which are three and zero respectively. $\mathcal{R}_+(C)$ being three satisfies the condition of $\mathcal{R}_+(C) \ge n-1$. Thus, $\rho_c^h = 1$, which means that the comment in Table 1 has positive sentiment. For the same comment $\mathcal{R}_-(C) < n-1$ and so $\eta_c^h = -1$, which signifies that the comment has no negative sentiment.

2.4. Sentiment detection using `SentiStrength`

We apply `SentiStrength` to determine the sentiments expressed in the issue comments in Group-1 of the "Gold Standard" dataset. Sentiment analysis using `SentiStrength` on a given piece of text (e.g., an issue comment) C computes a pair $\langle \rho_c, \eta_c \rangle$ of integers, where $+1 \le \rho_c \le +5$ and $-5 \le \eta_c \le -1$. Here, ρ_c and η_c respectively represent the positive and negative sentimental scores for the given text C. A given text C is considered to have positive sentiment if $\rho_c > +1$. Similarly, a text is held containing negative sentiment when $\eta_c < -1$. Besides, a text is considered sentimentally neutral when the sentimental scores for the text appear to be $\langle 1, -1 \rangle$.

Hence, for the pair $\langle \rho_c, \eta_c \rangle$ of sentimental scores for an issue comment C computed by `SentiStrength`, we compute another pair of integers $\langle \rho_c^t, \eta_c^t \rangle$ as follows:

$$\rho_c^t = \begin{cases} 1, & \text{if } \rho_c > +1. \\ 0, & \text{otherwise.} \end{cases} \quad \text{and} \quad \eta_c^t = \begin{cases} 1, & \text{if } \eta_c < -1. \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

Here, $\rho_c^t = 1$ signifies that the issue comment C has positive sentiment, and $\eta_c^t = 1$ implies that the issue comment C has negative sentiment.

We apply `SentiStrength` to compute sentimental scores for each of the issue comments in the Group-1 portion of the "Gold Standard" dataset and then for each issue comment C, we compute the pair $\langle \rho_c^t, \eta_c^t \rangle$, which represents the sentimental polarity scores for C.

2.5. Analysis and findings

For each of the 392 issue comments C in Group-1, we compare the sentimental polarity scores $\langle \rho_c^t, \eta_c^t \rangle$ produced from `SentiStrength` and the scores $\langle \rho_c^h, \eta_c^h \rangle$ computed using our approach described in

Section 2.3. We find a total of 151 comments, for which the $\langle \rho_c^t, \eta_c^t \rangle$ scores obtained from SentiStrength do not match with $\langle \rho_c^h, \eta_c^h \rangle$. This implies that for those 151 issue comments SentiStrength's computation of sentiments are probably incorrect.

Upon developing a solid understanding of the sentiment detection algorithm of SentiStrength, we then carefully go through all of those 151 issue comments to identify the reasons/difficulties, which mislead SentiStrength in its identification of sentiments in textual content. We identify 12 such difficulties. Before discussing the difficulties, we first briefly describe the highlights of SentiStrength's internal working mechanism to develop necessary context and background for the reader.

2.5.1. Insights into SentiStrength's internal algorithm

SentiStrength is a lexicon-based classifier that also uses additional (non-lexical) linguistic information and rules to detect sentiment in plain text written in English (Thelwall et al., 2012). SentiStrength maintains a dictionary of several lists of words and phrases as its key dictionaries to compute sentiments in texts. Among these lists, the *sentimental words list, list of booster words, list of phrases*, and *list of negations words* play a vital role in the computation of sentiments. The entries in all these lists except the list of negation words are pre-assigned with sentimental scores. The negation words in the fourth list are used to invert the sentimental polarity of a term when the term is located after a negation word in text.

For an input sentence, SentiStrength extracts individual words from the sentence and searches for each of the individual words in the *sentimental words list* to retrieve the corresponding sentimental scores. Similar search is made in the *list of booster words* to strengthen or weaken the sentimental scores. The *list of phrases* is used to distinguish groups of words as commonly used phrases. When such a phrase is identified, the sentimental score of the phrase overrides sentimental scores of the individual words, which constitute the phrase. The examples in Table 2 articulate how SentiStrength depends on the dictionary of lists for computing sentimental scores in plain texts.

2.5.2. Difficulties in automated sentiment analysis in software engineering

Table 3presents the number of times we found SentiStrength being misled by the 12 difficulties as discovered during manual investigation. It is evident in Table 3 that *domain-specific meanings of words* is the most prevalent among all the difficulties that are liable for low accuracy of the lexical approach of SentiStrength. However, not all the difficulties are specific to software engineering domain, rather some difficulties impact sentiment analysis in general (including software engineering) while a few are actually specific limitations of the tool SentiStrength. The right-most column in Table 3 indicates the scopes of the identified difficulties. We now describe 12 difficulties with illustrative examples.

(D_1) Domain-specific meanings of words: In a technical field, textual artifacts include many technical jargons, which have polarities in terms of dictionary meanings, but do not really express any sentiments in their technical context. For example, the words 'Super', 'Support', 'Value' and 'Resolve' are English words with known positive sentiment, whereas 'Dead', 'Block', 'Default', and 'Error' are known to have negative sentiment, but none of these words really bear any sentiment in software development artifacts.

Table 3
Frequencies of difficulties misleading sentiment analysis

Difficulties		Frequency (%)	Scope*
D_1:	Domain-specific meanings of words	123 (60.00)	SEDS
D_2:	Context-sensitive variations in meanings of words	35 (17.07)	SAG
D_3:	Misinterpretation of the letter 'X'	12 (05.85)	SEDS
D_4:	Sentimental words in copy-pasted content (e.g., code)	12 (05.85)	SEDS
D_5:	Difficulties in dealing with negations	08 (03.90)	SAG
D_6:	Missing sentimental words in dictionary	02 (00.97)	SAG
D_7:	Spelling errors mislead sentiment analysis	02 (00.97)	SAG
D_8:	Repetitive numeric characters considered sentimental	01 (00.49)	SST
D_9:	Wrong detection of proper nouns	01 (00.49)	SST
D_{10}:	Sentimental words in interrogative sentences	01 (00.49)	SST
D_{11}:	Difficulty in dealing with irony and sarcasm	01 (00.49)	SAG
D_{12}:	Hard to detect subtle expression of sentiments	07 (03.41)	SAG

*Here, SEDS = Software Engineering Domain Specific, SAG = Sentiment Analysis in General, SST = Specific to the SentiStrength Tool.

As SentiStrength was originally developed and trained for non-technical texts written in plain English, it identifies those words as sentimental words, which is incorrect in the context of a technical field such as software engineering. In the following comment from the "Gold Standard" dataset, SentiStrength considers 'Error' as negative sentimental word and detects 'Support' and 'Refresh' as positive sentimental words. Thus, it assigns both positive and negative sentimental scores to the comment, although the comment is sentimentally neutral.

> "This was probably fixed by WODEN-86 which introduced **support** for the curly brace syntax in the http location template. This JIRA can now be closed. This test case is now passing ... There are now 12 **errors** reported for Woden on this test caseregenerated the results in r480113. I'll have the W3C reports **refreshed** ." (Comment ID: 18059)

(D_2) Context-sensitive variations in meanings of words: Apart from domain-specific meanings of words, in natural language, some words have multiple meanings depending on the context in which they are used. For example, the word 'Like' expresses positive sentiment when it is used in a sentence such as *"I like you"*. On the other hand, that same word expresses no sentiment in the sentence *"I would like to be a sailor, said George Washington"*. Again, SentiStrength identifies the word 'Please' as positive sentimental word, although we find the word is used as neutral to express request in the training dataset. For example, in the comment below, the word 'Please' does not express any emotion.

> "Updated in 1.2 branch. David; **please** download and try 1.2 beta when it is released in a week or so.." (Comment ID: 4223)

Again, words that are considered inherently sentimental often do

Table 2
The role of the dictionary lists in SentiStrength's computation of sentimental scores in text

Sample	Sent. Scores		Dictionary	Explanation
sentence	ρ_c	η_c	list in use	
It's a *good* feature.	2	-1	Sentimental words	The sentimental score of the word *'good'* is pre-assigned to 02; so the sentence is assigned positive score 02.
It's a *very* good feature.	3	-1	Booster words	As booster word *'very'* is used before the sentimental word, the sentence is assigned a positive score 03.
It's *not* a good feature.	1	-1	Negations	Sentimental polarity of the sentimental word is inverted in here due to the use of the negation word *'not'* before sentimental word.
It's a *killer feature*.	2	-1	Phrases	*"killer feature"* is a phrase in the dictionary with positive score 02. Although the word *'kill'* carries negative sentiment, its effect is overridden by the sentimental score of the phrase.

not carry sentiments when used to express possibility and uncertainty. Distinguishing the context-sensitive meanings of such words is a big challenge for automated sentiment analysis in text and the lexical approach of SentiStrength also falls short in this regard.

For example, in the following issue comment, the sentimental word 'Nice' is used simply to express possibility regarding change of something, but SentiStrength incorrectly computes positive sentiment in the message.

"The change you want **would be nice** ; but is simply not possible. The form data ... Jakarta FileUpload library." (Comment ID: 51837)

Similarly, in the comment, the sentimental word 'Misuse' is used in a conditional sentence, which does not express any sentiment, but SentiStrength interprets otherwise.

"Added a couple of small points ... **if** anyone notices any **misuses** of the document formatting ..." (Comment ID: 2463)

(D_3) Misinterpretation of the letter 'X': In informal computer mediated chat, the letter 'X' is often used to mean an action of 'Kiss', which is a positive sentiment, and thus recorded in SentiStrength's dictionary. However, in technical domain, the letter is often used as a wildcard. For example, the sequence '1.4.x' in the following comment is used to indicate a collection of versions/releases.

"Integrated in Apache Wicket **1.4.x** ..." (Comment ID: 20748)

Since SentiStrength uses dot (.) as a delimiter to split a text into sentences, the 'x' is considered a one-word sentence and is misinterpreted to have expressed positive sentiment.

(D_4) Sentimental words in copy-pasted content (e.g., code): At commit, the developers often copy-paste code snippets, stack traces, URLs, and camel-case words (e.g., variable names) in their issue comment. Such copy-pasted contents often include sentimental words in the form of variable names and the like, which do not convey any sentiment of the committer, but SentiStrength detects those sentimental words and incorrectly associates those sentiments with the issue comment and the committer. Consider the following issue comment, which includes a copy-pasted stack trace.

"... Stack: [main] **FATAL** ... org.apache.xalan.templates .ElemTemplateElement. **resolve** PrefixTables ..." (Comment ID: 9485)

The words 'Fatal' and 'Resolve' (part of the camel case word 'resolvePrefixTables'), are positive and negative sentimental words respectively in the dictionary of SentiStrength's. Hence, SentiStrength detects both positive and negative sentiments in the issue comment, but the stack trace content certainly does not represent the sentiments of the developer/committer.

(D_5) Difficulties in dealing with negations: For automated sentiment detection, it is crucial to identify the presence of any negation term preceding a sentimental word, because the negation terms invert the polarity of the sentimental words. For example, the sentence *"I am not in good mood"* is equivalent to *"I am in bad mood"*. When the negation of the positive word 'Good' cannot be identified as equivalent to the negative word 'Bad', then detection of sentimental polarity goes wrong. The default configuration of SentiStrength enables it to detect negation of a sentimental word *only if* the negation term is placed *immediately before* the sentimental word. In all other cases, SentiStrength fails to detect negations correctly and often detects sentiments exactly opposite of what is expressed in the text. During our investigation, we find substantial instances where SentiStrength is misled by complex structural variations of negations present in the issue comments.

For example, in the following two comments, SentiStrength cannot detect negation, which are used before the word 'Bad' (in first comment) and 'Good' (in second comment) correctly and thus misclassified sentiments of those comments.

"I **haven't** seen any **bad** behavior. I was using open ssh to test this. I used the .... with open ssh to disconnect;" (Comment ID: 6688)

"3.0.0 has been released; closing ... I didn't change the jute - **don't** think this is a **good** idea; esp as also effects the ..... Andrew could you take a look at this one?" (Comment ID: 1725)

In addition, we find that SentiStrength is unable to recognized shortened forms of negations such as, "haven't", "havent", "hasn't", "hasnt", "shouldn't", "shouldnt", and "not" since these terms are not included in the dictionary.

(D_6) Missing sentimental words in dictionary: Since the lexical approach of SentiStrength is largely dependent on its dictionary of lists of words (as discussed in Section 2.5.1), the tool often fails to detect sentiments in some texts when the sentimental words used in the texts are absent in the dictionary. For example, the words 'Apology' and 'Oops' in the following two comments express negative sentiments, but SentiStrength cannot detect them since those words are not included in its dictionary.

"...This is indeed not an issue. My **apologies** ..." (Comment ID: 20729)

" **Oops** ; issue comment had wrong ticket number in it ..." (Comment ID: 36376)

(D_7) Spelling errors mislead sentiment analysis: Misspelled words are common in informal text, and the writer often deliberately misspells words to express intense sentiments. For example, the misspelled word 'Happpy' expresses more happiness than the correctly spelled word 'Happy'. Although SentiStrength can detect some of such intensified sentiments from such misspelled sentimental words, its ability is limited to only those intentional spelling errors where repetition of certain letters occur in a sentimental word. Most other types (unintentional) of misspelling of sentimental words cause SentiStrength fail to find those words in its dictionary and consequently lead to incorrect computation of sentiments. For example, the word 'Unfortunately' was misspelled as 'Unforunatly' in an issue comment (comment ID: 11978) and "I'll" was written as 'ill' in another (comment ID: 927). SentiStrength's detection sentiments in both of these comments are found incorrect.

(D_8) Repetitive characters considered sentimental: As described before, SentiStrength detects higher intensity of sentiments by considering deliberately misspelled sentimental word with repetitive letters. The tool also uses the same strategy for the same purpose by taking into account repetitive characters intentionally typed in words that are not necessarily sentimental by themselves. If anybody writes *"I am goooing to watch movie"* instead of *"I am going to watch movie"*, then the former sentence is considered positively sentimental due to emphasis on the word 'Going' by repetition of the letter 'O' for three times.

However, this strategy also misguides SentiStrength in dealing with some numeric values. For example, in the following comment, SentiStrength incorrectly identifies the number '20001113' as a positive sentimental word encountering repetition of the digits '0' and '1'.

"See bug 5694 for the ... **20001113** /introduction.html ... Zip file with test case (java source and XML docs) 1. Do you use deferred DOM? 2. Can you try to run it against Xerces2 beta4 (or the latest code in CVS?) 3. Can you provide a sample file? Thank you." (Comment ID: 6447)

(D_9) Incorrect detection of proper nouns: A proper noun can

rightly be considered neutral in sentiment. SentiStrength detects a word starting with a capital letter as a proper noun, when the word is located in the middle or end of a sentence. Unfortunately, grammar rules are often ignored in informal text and thus, sentimental words placed in the middle or end of a sentence often end up starting with a capital letter, which cause SentiStrength mistakenly disregard the sentiments in those sentimental words. The following issue comment is an example of such a case, where the sentimental word 'Sorry' starting with a capital letter is placed in the middle of the sentence and SentiStrength erroneously considers 'Sorry' as a neutral proper noun.

> "Cool. Thanks for considering my bug report! ... About the title of the bug; in the description; I put: **Sorry** for the vague ticket title. I don't want to make presumptions about the issue ... work for passwords." (Comment ID: 76385)

However, the older *Windows version* of SentiStrength does not have this shortcoming.

(D_{10}) Sentimental words in interrogative sentences: Typically, negative sentimental words in interrogative sentences (i.e., in questions) either do not express any sentiment or at least weaken the intensity of sentiment (Thelwall et al., 2012). However, we have found instances where SentiStrength fails to correctly interpret the sentimental polarities of such interrogative sentences. For example, SentiStrength incorrectly identifies negative sentiment in the comment below, although the comment merely includes a question expressing no negative sentiment as indicated by the human raters.

> "... Did I submit something wrong or duplicate? ..." (Comment ID: 24246)

(D_{11}) Difficulty in dealing with irony and sarcasm: Automatic interpretation of irony in text written in natural language is very challenging, and SentiStrength also often fails to detect sentiments from texts, which express irony and sarcasm (Thelwall et al., 2012). For example, due to the presence of the positive sentimental words "Dear God!" in the comment below, SentiStrength detects positive sentiment in the sentence, although the comment poster used it in a sarcastic manner and expressed negative sentiment only.

> "The other precedences are OK; as far as I can tell ... **'zzz' ; Dear God!** You mean the intent here is ... gotta confess I just saw the pattern and jumped to conclusions; hadn't examined the code at all. But you've just made the job tougher ...?" (Comment ID: 61559)

(D_{12}) Hard to detect subtle expression of sentiments: Text written in natural language can express sentiments without using any inherently sentimental words. The lexical approach of SentiStrength fails to identify sentiments in such a text due to its high dependency on the dictionary of lists of words, and not being able to properly capture sentence structure and semantic meanings. Consider the following issue comment, which was labeled with negative sentiment by three human raters although there is no sentimental words in it. Without surprise, SentiStrength incorrectly interprets it as a sentimentally neutral text.

> "Brian; I understand what you say and specification about 'serialization' in XSLT not 'indenting'. As I saied before; indenting is just the thing that we easily see the structure and data of XML document. Xalan output is not easy to see that. The last; I think the example of non-whitespace characters is no relationship to indenting. non-whitespace characters must not be stripped; but whitespace characters could be stripped. Regards; Tetsuya Yoshida." (Comment ID: 10134)

3. Leveraging automated sentiment analysis

We address the challenges identified from our exploratory study as described in Section 2.5.2 and develop a tool particularly crafted for application in the software engineering domain. We call our tool SentiStrength-SE, which is built on top of the original SentiStrength. We now describe how we mitigate the identified difficulties in developing SentiStrength-SE for improved sentiment analysis in textual artifacts in software engineering.

3.1. Creating a new domain dictionary for software engineering

As reported in Table 3, our exploratory study (Section 2.5.2) found the domain-specific challenges (difficulties D_1, D_3, D_6) as the most impeding factors against sentiment analysis in software engineering text. Hence, the accuracy of sentiment analysis can be improved by adopting a domain-specific dictionary (Godbole et al., 2007; Hu and Liu, 2004; Qiu et al., 2009). We, therefore, first create a domain dictionary for software engineering text to address the issues with domain difficulty.

Fig. 1depicts the steps/actions taken to develop the domain dictionary for software engineering text. We collect a large dataset used in the work of (Islam and Zibran, 2016b). This dataset consists of 490 thousand commit messages drawn from 50 open-source projects from GitHub. Using the Stanford NLP tool (StanfordCoreNLP, last access: June 2018), we extract a set of lemmatized forms of all the words in the commit messages, which is denoted as M_w.

To identify the emotional words from the set M_w, we exploit

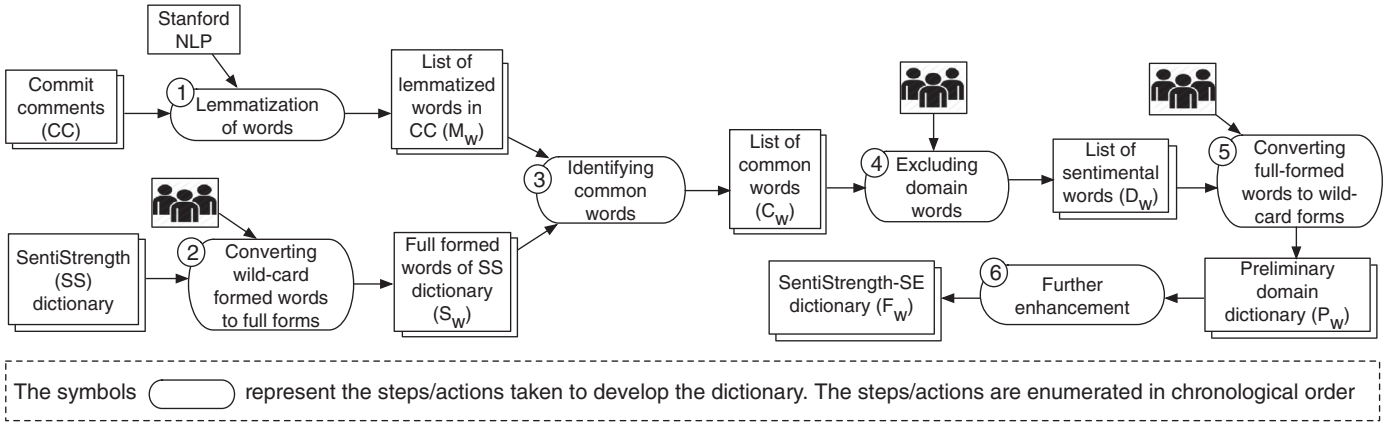

Fig. 1. Steps to create the domain dictionary for SentiStrength-SE

SentiStrength's existing dictionary. We choose SentiStrength's existing dictionary as the basis for our new one, because, in a recent study (Islam and Zibran, 2017a), SentiStrength's dictionary building method is found superior to other approaches (AFINN (Nielsen, 2011), MPQA (Wilson et al., 2009), and VADER (Hutto and Gilbert, 2014)) for the creation of software engineering domain-specific dictionaries. We identify those words in SentiStrength's original dictionary, which have wild-card forms (i.e., words that have symbol * as suffix) and transform them to their corresponding lemmatized forms using the AFINN (Nielsen, 2011) dictionary. For example, the entry 'Amaz*' in SentiStrength dictionary is transformed to the words 'Amaze', 'Amazed', 'Amazes' and 'Amazing' as those are found as emotional words in the AFINN dictionary corresponding to that particular entry. The are mainly two reasons for using the AFINN dictionary: (i) the dictionary is very similar to the SentiStrength dictionary as both use the same numeric scale to express sentimental polarities of words, (ii) the AFINN dictionary is also widely used in many other studies (Riloff et al., 2013; Gan and Yu, 2015; Koto and Adriani, 2015; Islam and Zibran, 2017a). If any wild-card form word is not found in AFINN dictionary, we use our own wisdom to convert that word to its lemmatized forms. Thus, by converting all short forms words to full forms and combining those with remaining words in SentiStrength dictionary we obtain a set of words S_w. Then, we distinguish a set C_w of words such that $C_w = M_w \cap S_w$. The set C_w ends up containing 716 words, which represent an initial sentimental vocabulary pertinent to the software engineering domain.

We recognize that some of these 716 words are simply software engineering domain-specific technical terms expressing no sentiments in software engineering context, which otherwise would express emotions when interpreted in a non-technical area such as social networking. There also remain other words such as *'Decrease', 'Eliminate'* and *'Insufficient'*, which are unlikely to carry sentiments in the software engineering domain. We, therefore, engage three human raters (enumerated as A, B, C) to independently identify these non-sentimental domain words in C_w. Each of these three human raters are computer science graduate students having at least three years of software development experience. A human rater annotates a word as neutral if the word appears to him/her as highly unlikely to express any sentiment when interpreted in the software engineering domain.

In Table 4, we present sentiment-wise percentage of cases where the human raters disagree pair-wise. We also measure the degree of inter-rater agreement in terms of *Fleiss-κ* (Fleiss, 1971) value. The obtained *Fleiss-κ* value 0.739 signifies substantial agreement among the independent raters.

We consider a word as a neutral domain word when two of the three raters identify the word as neutral. Thus, 216 words are identified as neutral domain words, which we exclude from the set C_w resulting in another set D_w of the remaining 500 words. Such neutralization of words for a particular domain is also suggested in several studies (Pletea et al., 2014; Tourani et al., 2014; Islam and Zibran, 2016a; 2016b) in the literature.

Next, we adjust the words in D_w by reverting them to their wild-card forms (if available) to comply with SentiStrength's original dictionary. This new set of words is called as preliminary domain dictionary (P_w), which has 167 positively and 310 negatively polarized words. This preliminary dictionary is further enhanced according to the description below to create the SentiStrength-SE dictionary (F_w).

Table 4
Inter-rater disagreements in interpretation of sentiments

	Disagreements between human raters		
Sentimental polarity	A, B	B, C	C, A
Positive	11.81%	19.68%	17.32%
Negative	08.62%	10.19%	09.41%
Neutral	18.13%	11.81%	15.69%

3.1.1. Further enhancements to the preliminary domain dictionary

We further extend the newly developed preliminary dictionary in the light of our observations during the exploratory study described in Section 2.

Extension with new sentimental words and negations: During our exploratory study, we find several informal sentimental words such as, 'Woops', 'Uhh', 'Oops' and 'zzz', which are not included in the original dictionary. The formal word 'Apology' is also missing from the dictionary. We have added to the dictionary of our SentiStrength-SE all these missing words as sentimental terms with appropriate sentimental polarities, which mitigate the difficulty D_6.

In addition, we also add to the dictionary the missing shortened from of negation words as mentioned in the discussion of difficulty D_5 in Section 2.5.2.

Discarding the letter 'X' from dictionary: We exclude the letter 'X' from our domain dictionary of SentiStrength-SE to save lexical sentiment analysis from the difficulty D_3 as described in Section 2.5.2.

3.2. Inclusion of heuristics in computation of sentiments

While the creation of the new domain dictionary is a vital step towards automated sentiment analysis in software engineering text, we realize that the computations for sentiment detection also need improvements. Thus, in the implementation of our domain-specific SentiStrength-SE, we incorporate a number of heuristics in the computation, which we describe below.

3.2.1. Addition of contextual sense to minimize ambiguity

Recall that, in the creation of our initial domain dictionary, we neutralized 216 words on the basis of the judgements from three independent human raters. However, the neutralization of words is not always appropriate. For example, in the software engineering domain, the word 'Fault' typically indicates a program error and expresses neutral sentiment. However, the same word can also convey negative sentiment as found in the following comment.

> "As WING ... **My fault:** I cannot reproduce after holidays ... I might add that one; too" the word 'Fault' expresses negative sentiment of the comment poster." (Comment ID: 4694)

Again, the word 'Like' expresses positive sentiment if it is used as *"I like", "We like", "He likes"*, and *"They like"*. In the most other cases the word 'Like' is used as *preposition* or *subordinating conjunction* and the word can safely be considered sentimentally neutral. For example, the following comment used the word 'Like' without expressing any sentiment.

> "Looks **like** a user issue to me ..." (Comment ID: 40844)

We can observe from the above examples that some of the 216 neutralized words can actually express sentiments when those are preceded by *pronouns* referring to a person or a group of persons, e.g., 'I', 'We', 'My', 'He', 'She', 'You' and possessive pronouns such as 'My' and 'Your'. This contextual information is taken into account in SentiStrength-SE to appropriately deal with the contextual use of those words in software engineering field to minimize the difficulties D_1 and D_2. The complete list of such words is given in the SentiStrength-SE dictionary file named 'ModifiedTermsLookupTable', which are also vetted by the three raters. Note that to determine the Part-Of-Speech (POS) of words in sentences, we apply the Stanford POS tagger (StanfordCoreNLP, last access: June 2018).

3.2.2. Bringing neutralizers in effect

Our observations from the exploratory study (as presented in Section 2) reveal that sentiment of a word can be neutralized if that word is preceded by any of the neutralizer words such as, 'Would',

'Could', 'Should', and 'Might'. For example in the sentence "*It would be good if the test could be completed soon*" the positive sentimental word 'Good' does not express any sentiment as neutralized by the preceding word 'Would'. We add a method in SentiStrength-SE to enable it correctly detect uses of such neutralizer words in sentences to be more accurate in sentiments detection. This helps in minimizing the difficulty D_2 described before.

3.2.3. Integration of a preprocessing phase

To minimize the difficulties D_4, D_7, D_8, and D_9 (as described in Section 2.5.2), we include a preprocessing phase to SentiStrength-SE as its integral part. Before computation for any given input text, SentiStrength-SE applies this preprocessing phase to filter out numeric characters and certain copy-pasted contents such as code snippets, URLs and stack traces. To locate code snippets, URL's and stack traces in text, we use regular expressions similar to the approach proposed by Bettenburg et al. (2011). In addition, a spellchecker (Jazzy- The Java Open Source Spell Checker, 0000) is also included to deal with the difficulty D_7 in identifying and rectifying misspelled English words. Spell checking also complements our regular expression based method in approximate identification of identifier names in code snippets.

To mitigate the difficulty D_9 in particular, the preprocessing phase also converts all the letters of a comment to small letters. However, converting all the letters to small letters can also cause failure of the detection of the proper nouns such as the names of developers and systems, which is also important as discussed in the description of difficulty D_9 in Section 2.5.2. From our exploratory study, we have observed that the developers typically mention their colleagues' names in comments immediately after some sort of salutation words such as 'Dear', 'Hi', 'Hello', 'Hellow' or after the character '@'. Hence, in addition to converting all letters to lower case, the preprocessing phase also discards those words, which are placed immediately after any of those salutation words or the character '@'. In addition, SentiStrength-SE maintains the flexibility to allow the user to instruct the tool to consider any particular words as neutral in sentiment, in case an inherently sentimental word must be recognized as proper noun, for example, to deal with the situation where a sentimental word is used as a system's name.

3.2.4. Parameter configuration for better handling of negations

We carefully set a number of configuration parameters as defaults to our SentiStrength-SE tool as shown in Fig. 2. This default configuration of SentiStrength-SE is different from that of the original SentiStrength. Particularly, to mitigate the difficulty D_5 in dealing with negations, the negation's configuration parameter marked with a black rectangle in Fig. 2 is set to five in SentiStrength-SE, which enables the tool detecting negations over a larger range of proximity allowing zero to five intervening words between a negation and a sentimental word, as was also suggested in a previous study (Hu and Liu, 2004).

4. Empirical evaluation of SentiStrength-SE

While making the design and tuning decisions to develop SentiStrength-SE, we remain careful about the possibility that the application of a particular heuristic for improvement in one area might have side-effects causing performance degradation in another criteria. We empirically evaluate our domain-specific techniques and the accuracy of domain-specific SentiStrength-SE in several phases around seven research questions.

Dataset: For empirical evaluation of our SentiStrength-SE, we use the 5,600 issue comments in Group-2 and Group-3 of the "Gold Standard" dataset introduced in Section 2.1. The *ground-truth* about the sentimental polarities of those issue comments are determined based on the manual evaluations by human raters as described in Section 2.3. Before conducting evalution, we present textual characteristics of Group-2 and Group-3 datasets in Table 5, which indicates no substantial differences in the characteristics of the two datasets.

Metrics: The accuracy of sentiment analysis is measured in terms of *precision, recall*, and *F-score* computed for each of the three sentimental polarities (i.e., positivity, negativity and neutrality). Given a set S of textual contents, *precision* ($\wp$), *recall* ($\Re$), and *F-score* (Ǝ) for a particular sentimental polarity e is calculated as follows:

$$\wp = \frac{|S_e \cap S'_e|}{|S'_e|}, \qquad \Re = \frac{|S_e \cap S'_e|}{|S_e|}, \qquad \text{Ǝ} = \frac{2 \times \wp \times \Re}{\wp + \Re}$$

where S_e represents the set of texts having sentimental polarity e, and S'_e denotes the set of texts that are detected (by tool) to have the sentimental polarity e.

Statistical test of significance: We apply non-parametric *McNemar's* test (Dietterich, 1998) to verify the statistical significance in the difference of the results obtained by two tools, say $\mathcal{T}_a$ and $\mathcal{T}_b$. As the non-parametric test does not require normal distribution of data, this test suits well for our purpose. We perform a *McNemar's* test on 2×2 contingency table derived from the results obtained from tools $\mathcal{T}_a$ and $\mathcal{T}_b$. The structure of such a contingency table is shown in Table 6.

Let, F_a and F_b denote the sets of misclassified comments by $\mathcal{T}_a$ and $\mathcal{T}_b$ respectively. In the contingency table (Table 6), n_{00} represents the number of issue comments misclassified by both $\mathcal{T}_a$ and $\mathcal{T}_b$ (i.e., $n_{00} = |F_a \cap F_b|$), n_{01} represents the number of comments misclassified by $\mathcal{T}_b$ but not by $\mathcal{T}_a$ (i.e., $n_{01} = |F_b - F_a|$), n_{10} represents the number of comments misclassified by $\mathcal{T}_a$ but not by $\mathcal{T}_b$ (i.e., $n_{01} = |F_a - F_b|$), and n_{11} represents the number of comments correctly classified by both the tools. Let, S denote the set of all the issue comments correctly classified according to the ground-truth. Thus, $n_{11} = S - (F_a \cup F_b)$. The superiority of tool $\mathcal{T}_b$ over the tool $\mathcal{T}_a$ is observed, if $n_{10} > n_{01}$. Otherwise, $\mathcal{T}_a$ is considered superior if $n_{01} > n_{10}$. Such observed superiority is considered statistically significant, if the *p-value* obtained from a *McNemar's* test is less than a pre-specified significance level α. In our work, we set $\alpha = 0.001$, which a reasonable setup widely used in the literature.

4.1. Head-to-head comparison using a benchmark dataset

We compare our software engineering domain-specific SentiStrength-SE with the original SentiStrength (Thelwall et al., 2012) tool and two other toolkits NLTK (NLTK, 0000) and Stanford NLP (Socher et al., 2013b). To the best of our knowledge, these are the only *domain independent* tools/toolkits attempted in the past for sentiment analysis in software engineering text (Pletea et al., 2014; Ortu et al., 2015; Novielli et al., 2015; Rahman et al., 2015). In particular, we address the following research question:

> **RQ1:** *Does our domain-specific SentiStrength-SE outperform the existing domain independent tools for sentiment analysis in software engineering text?*

We write a Python script to import NLTK sentiment analysis package (Sinha, 2016; NLTK, 0000) and run it on texts to determine the sentimental polarities of those. NLTK determines the probability of positivity, negativity and neutrality of a text. In addition, its also provides a compound value C_v, which ranges between -1 to +1. A text contains positive sentiments if $C_v > 0$, a text will have negative sentiments if $C_v < 0$. Otherwise, a text is considered sentimentally neutral when $C_v = 0$. Similar procedure is also followed in another study (Sinha, 2016) to determine sentiments of texts using NLTK.

We develop a Java program using the JAR of the Stanford NLP tool to apply it on texts to determine their sentimental polarities. For a text Stanford NLP provides a sentiment score S_v between zero to four where $0 \le S_v \le 1$ indicates negative sentiment, $3 \le S_v \le 4$ indicates positive sentiment and $S_v = 2$ indicates neutral sentiment of the text (Socher et al., 2013a).

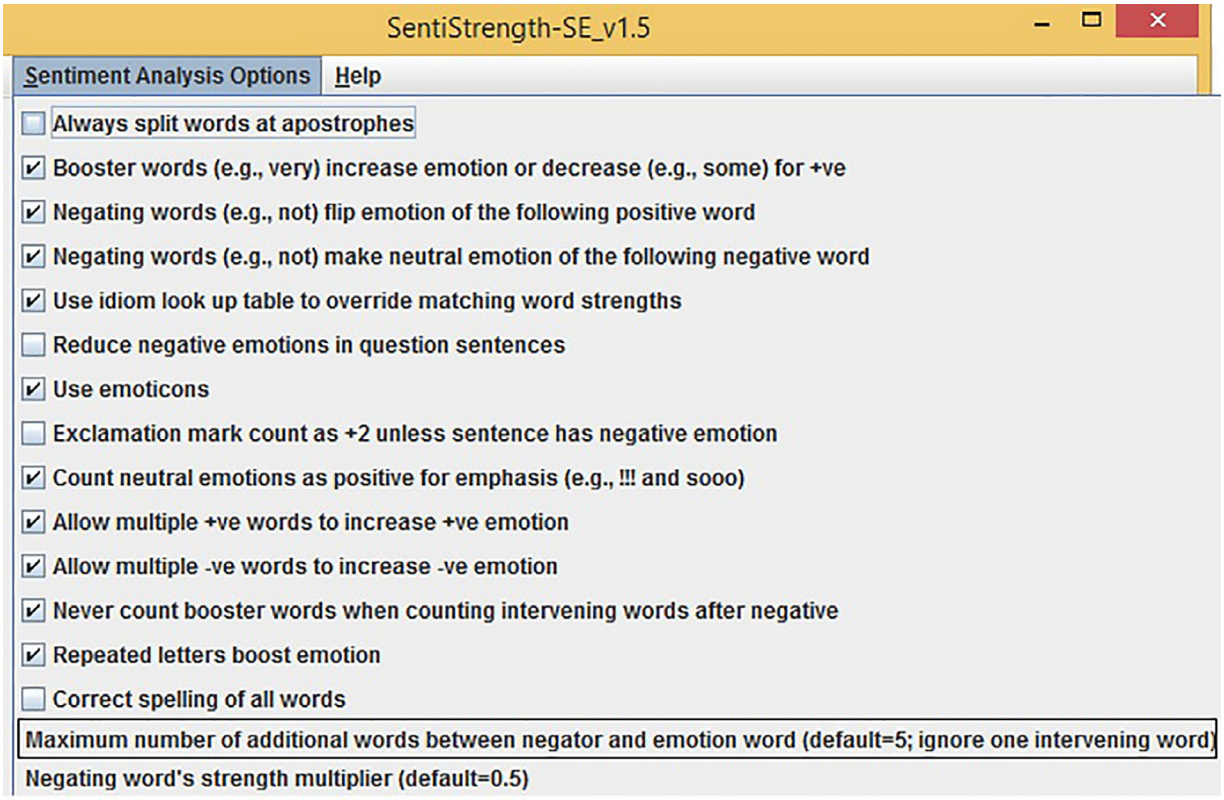

Fig. 2. Default configuration of parameters in our `SentiStrength-SE`

Table 5
Textual characteristics of the words in the datasets

Datasets	Number of distinct words	Complexity factor (Lexical density)	Number of sentences	Average sentence length (in words)	Number of sentences per comment
Group-2	5,295	21.00%	5,671	8.23	3.54
Group-3	5,527	24.30%	4,000	8.26	1.00

Table 6
Structure of 2 × 2 contingency matrix of McNemar's test for tools $\mathcal{T}_a$ and $\mathcal{T}_b$

# of comments misclassified by both $\mathcal{T}_a$ and $\mathcal{T}_b$	n_{00}	n_{01}	# of comments misclassified by $\mathcal{T}_b$ but not by $\mathcal{T}_a$
# of comments misclassified by $\mathcal{T}_a$ but not by $\mathcal{T}_b$	n_{10}	n_{11}	# of comments correctly classified by both $\mathcal{T}_a$ and $\mathcal{T}_b$

We separately operate each of the selected tools and our `SentiStrength-SE` on the Group-2 and Group-3 portions of the "Gold Standard" dataset. Recall that Group-2 and Group-3 datasets contain 1,600 and 4,000 issue comments respectively. For each of the three sentimental polarities (i.e., positivity, negativity, and neutrality), we compare the tools' outcome with the ground-truth and separately compute precision, recall, and F-score for all the tools with respect to each dataset. Table 7 presents the precision ($\wp$), recall ($\Re$), and F-score ($\digamma$) of all the tools in the detection of positive, negative and neutral sentiments, and also the average over all these three sentimental polarities. The highest metric values are highlighted in bold.

Notice that for the Group-2 dataset, our `SentiStrength-SE` consistently achieves the highest precision, recall and F-scored compared to the rest other tools.

For the Group-3 dataset, `SentiStrength-SE` achieves the highest precision and F-score in detecting negative sentiments and it achieves the highest recall and F-score in the detection of neutral sentiments. In those few cases, where `SentiStrength-SE` does not achieve the best results, it remains at the second best or marginally close to the best. In the detection of positive sentiments in Group-3 dataset, `Satnford NLP` achieves the highest precision and recall, where our `SentiStrength-SE` yields the second best results. Similarly, the original `SentiStrength` achives the highest recall in the detection of negative sentiments in Group-3 dataset, and again our `SentiStrength-SE` obtains the second best result. The highest precision (91.28% achieved by the original `SentiStrength`) for neutral sentiments is only 0.64%

Table 7
Head-to-head comparison of performances of the four tools/toolkits

Data	Sentiments	Met.	Senti-Strength-SE	Senti-Strength	NLTK	Stanford NLP
Group-2	Positive	$\wp$	**88.86%**	74.48%	69.47%	79.77%
		$\Re$	**98.81%**	**98.81%**	81.55%	71.67%
		$\digamma$	**93.57%**	84.93%	75.0%	75.50%
	Negative	$\wp$	**53.42%**	28.22%	40.46%	13.28%
		$\Re$	**97.66%**	**97.66%**	54.69%	88.28%
		$\digamma$	**69.06%**	43.78%	46.51%	23.08%
	Neutral	$\wp$	**98.14%**	96.83%	69.53%	63.70%
		$\Re$	**83.00%**	52.42%	50.86%	25.57%
		$\digamma$	**89.94%**	68.01%	58.75%	36.49%
Group-3	Positive	$\wp$	41.80%	31.69%	20.32%	**69.47%**
		$\Re$	82.04%	**87.79%**	86.33%	81.55%
		$\digamma$	55.39%	46.58%	32.89%	**75.03%**
	Negative	$\wp$	**68.61%**	47.61%	50.65%	40.46%
		$\Re$	71.00%	**78.40%**	70.24%	54.69%
		$\digamma$	**69.78%**	59.25%	58.86%	46.51%
	Neutral	$\wp$	90.64%	**91.28%**	91.17%	69.53%
		$\Re$	**80.05%**	56.16%	45.78%	50.86%
		$\digamma$	**85.02%**	69.54%	60.96%	58.74%
Overall average accuracy		$\wp$	**73.58%**	61.69%	56.93%	56.04%
		$\Re$	**85.43%**	78.54%	64.91%	62.10%
		$\digamma$	**79.06%**	62.02%	55.50%	52.56%

Table 8
Contingency matrix of McNemar's test in comparison between the original SentiStrength and SentiStrength-SE

# of comments misclassified by both $\mathcal{T}_a$ and $\mathcal{T}_b$	748	365	# of comments misclassified by $\mathcal{T}_b$ but not by $\mathcal{T}_a$
# of comments misclassified by $\mathcal{T}_a$ but not by $\mathcal{T}_b$	**1,527**	2,924	# of comments correctly classified by both $\mathcal{T}_a$ and $\mathcal{T}_b$

Here, $\mathcal{T}_a$ = original SentiStrength and $\mathcal{T}_b$ = SentiStrength-SE

higher than that of our SentiStrength-SE.

Thus, if we consider the overall average accuracy, as presented at the bottom of the table, it becomes evident that our SentiStrength-SE performs the best, followed by the original SentiStrength and NLTK. Notice that the overall precision, recall and F-score of our SentiStrength-SE are substantially higher than those of the second-best performing tool (i.e., the original SentiStrength).

To verify whether the observed performance difference between SentiStrength-SE and the original SentiStrength is statistically significant, we perform a *McNemar's* test between the results of these two tools. For both datasets Group-2 and Group-3, we compute n_{00}, n_{01}, n_{10} and n_{11} according to their specifications described in Table 6. In Table 8, we present the contingency matrix computed for the *McNemar's* test. We observe superiority of $\mathcal{T}_b$ (SentiStrength-SE) in the contingency table as $n_{10} > n_{01}$. According to the p-value ($p = 2.2 \times 10^{-16}$, $p < \alpha$) obtained from the test, the observed difference in the superior performance of SentiStrength-SE is statistically significant. Based on these observations and statistical test, we now derive the answer to the research question RQ1 as follows:

> **Ans. to RQ1:** *In the detection of sentiments in software engineering text, our domain-specific SentiStrength-SEsignificantly outperforms the domain independent NLTK, Stanford NLP, and the original SentiStrength.*

4.2. Comparison with respect to human raters' disagreements

Recall that the issue comments in the "Gold Standard" dataset are annotated with sentiments as identified by independent human raters. There are disagreements among human raters in the identification of sentiments in some issue comments. While humans disagree about sentiments in some issue comments, it is likely that the automated tools will also produce different outcomes resulting in varied precision and recall.

We, therefore, investigate to what extend the agreements and disagreements of annotations among human raters cause deviation of results of the head-to-head comparison of tools as described in the previous section. Particularly, we want to verify whether our domain-specific SentiStrength-SE still outperforms the other domain-independent tools when the rater' agreements and disagreements are taken into account. Here, we address the following research question:

> **RQ2**: *Does the accuracy of SentiStrength-SElargely vary compared to its domain independent counterparts when the agreements and disagreements among human raters are taken into account?*

For this investigation, we use the Group-2 dataset where each issue comment was independently annotated by three human raters. We distinguish two sets of issue comments from this Group-2 dataset.

(i) *Set-A*: containing those issue comments for which all the three human raters agreed on the sentiments expressed in those comments. This set contains 1,210 issue comments.
(ii) *Set-B*: consisting of those issue comments for which two of the three raters agreed on the sentiments expressed in those comments. This set contains 357 issue comments.

We formulate the following null and alternative hypotheses to determine the statistical significance of improved performances of the best tool.

Null hypothesis-1 (H_o^1): There is no significant difference in the performance of SentiStrength-SE compared to the other tools in sentiment detection in the issue comments in *Set-A*.

Alternative hypothesis-1 (H_a^1): There exist significant differences in the performance of SentiStrength-SE compared to the other tools in sentiment detection in the issue comments in *Set-A*.

Null hypothesis-2 (H_o^2): There is no significant difference in the performance of SentiStrength-SE compared to the other tools in sentiment detection in the issue comments in *Set-B*.

Alternative hypothesis-2 (H_a^2): There exist significant differences in the performance of SentiStrength-SE compared to the other tools in sentiment detection in the issue comments in *Set-B*.

We now examine whether these hypotheses hold true with respect to the four tools we compare. In the similar way as of the head-to-head comparison described in the previous section, we separately run all the four tools including our SentiStrength-SE on *Set-A* and *Set-B* issue comments. For each of the three sentimental polarities (i.e., positivity, negativity, and neutrality), we compute precision ($\wp$), recall ($\Re$), and F-score ($\exists$) for each of the tools separately over the issue comments in both *Set-A* and *Set-B*.

For the issue comments in both *Set-A* and *Set-B*, the best tool *must* exhibit the significantly improved performances compared to other tools. For testing our hypotheses, in each of *Set-A* and *Set-B* datasets, we first identify the two tools producing better results among all the four tools. Then, we examine if there is any significant difference in the performances of the best tool and the second best one. If there exist significant differences *between* the accuracies of the top performing two tools, discovering such would suffice for demonstrating the existence of significant difference of the best performing tool against the other tools.

Table 9presents the metrics' values of all the tools in the detection of positive, negative and neutral sentiments for each set of the issue comments. As seen in Table 9, for the issue comments in *Set-A*, SentiStrength-SE consistently achieves the highest precision, recall and F-score in the detection of positive, negative and neutral sentiments.

Table 9
Comparison of tools' accuracies for *Set-A* and *Set-B* issue comments

Data	Sentiments	Met.	Senti-Strength-SE	Senti-Strength	NLTK	Stanford NLP
Set-A	Positive	$\wp$	**90.15%**	70.46%	67.9%	83.39%
		$\Re$	**95.33%**	91.20%	61.58%	76.66%
		$\exists$	**92.67%**	79.50%	64.17%	79.89%
	Negative	$\wp$	**53.66%**	28.17%	49.45%	9.66%
		$\Re$	**100.00%**	**100.00%**	54.55%	86.36%
		$\exists$	**69.84%**	43.96%	51.87%	17.38%
	Neutral	$\wp$	**99.44%**	99.43%	77.00%	67.61%
		$\Re$	**91.15%**	58.84%	54.08%	28.40%
		$\exists$	**95.12%**	73.93%	63.53%	40.00%
	Overall average accuracy	$\wp$	**81.08%**	66.02%	64.78%	53.55%
		$\Re$	**95.49%**	83.35%	56.74%	63.81%
		$\exists$	**85.88%**	65.79%	59.86%	45.76%
Set-B	Positive	$\wp$	**72.48%**	63.64%	67.32%	64.91%
		$\Re$	96.89%	**97.92%**	70.46%	57.13%
		$\exists$	**82.92%**	77.14%	68.86%	60.98%
	Negative	$\wp$	**51.75%**	21.63%	50.74%	20.83%
		$\Re$	**96.72%**	60.65%	55.73%	90.16%
		$\exists$	**67.42%**	31.89%	53.12%	33.85%
	Neutral	$\wp$	83.92%	**86.21%**	38.24%	38.88%
		$\Re$	**40.87%**	21.74%	33.91%	12.17%
		$\exists$	**54.97%**	34.72%	35.94%	18.54%
	Overall average accuracy	$\wp$	**69.38%**	57.16%	52.10%	41.54%
		$\Re$	**78.16%**	60.10%	53.37%	53.15%
		$\exists$	**68.44%**	58.59%	52.72%	37.79%

Table 10
Contingency matrix of McNemar's test between the accuracies of SentiStrength-SE and the original SentiStrength in *Set-A* dataset

# of comments misclassified by both $\mathcal{T}_a$ and $\mathcal{T}_b$	84	22	# of comments misclassified by $\mathcal{T}_b$ but not by $\mathcal{T}_a$
# of comments misclassified by $\mathcal{T}_a$ but not by $\mathcal{T}_b$	**511**	593	# of comments correctly classified by both $\mathcal{T}_a$ and $\mathcal{T}_b$

Here, $\mathcal{T}_a$ = original SentiStrength and $\mathcal{T}_b$ = SentiStrength-SE

The overall average accuracies indicate that the original SentiStrength achieves the second best accuracies in the *Set-A* dataset.

Now, we perform a *McNemar's* test between the accuracies of SentiStrength-SE and the original SentiStrength for the issue comments in *Set-A* dataset. The contingency table for the test is presented in Table 10. According to the contingency table, SentiStrength-SE ($\mathcal{T}_b$) performs better as $n_{10} > n_{01}$. The performance difference is found to be statistically significant with $p = 2.2 \times 10^{-16}$ and $p < \alpha$. Thus, the *McNemar's* test rejects our first null hypothesis (H_0^1). Therefore, the first alternative hypothesis (H_a^1) holds true.

Again, as seen in Table 9, for the issue comments in *Set-B*, SentiStrength-SE achieves the highest F-score in detecting sentiments of all the three polarities. The precision and recall of SentiStrength-SE is also the highest in all cases except for only two. The recall of SentiStrength-SE for positive sentiments is 96.89%, which is the second best and only 1.03% lower than the highest. Similarly, the precision of our SentiStrength-SE in detecting neutral sentiments is 83.92%, which is also next to the best and only 2.29% lower than the best. Also, with respect to the overall average accuracies, SentiStrength can be considered to have achieved the second best performance for the *Set-B* dataset.

Similar to the *Set-A* dataset, for the *Set-B*, we carry out a *McNemar's* test between the accuracies of SentiStrength-SE and the original SentiStrength to determine whether or not there is any statistical significant differences between the performances of these two tools. The contingency matrix for the test is presented in Table 11. According to the contingency matrix, SentiStrength-SE ($\mathcal{T}_b$) outperforms the original SentiStrength ($\mathcal{T}_a$) with $n_{10} > n_{01}$. The *McNemar's* test with the contingency matrix of Table 11 obtains $p = 2.2 \times 10^{-16}$ and thus $p < \alpha$. Thus, our second null hypothesis (H_0^2) is rejected and the second alternative hypothesis (H_a^2) holds true, which indicates that the superior performance of SentiStrength-SE over the original SentiStrength is statistically significant for the issue comments in the *Set-B* dataset.

Thus, for both the *Set-A* and *Set-B* datasets, SentiStrength-SE significantly outperforms the next best performer SentiStrength. Based on our observations and results from the statistical tests over both the *Set-A* and *Set-B* datasets, we now derive the answer to the research question RQ2 as follows:

Ans. to RQ2: *When the agreements and disagreements among human raters are taken into account, our domain-specific SentiStrength-SEstill maintains significantly superior (compared to its domain independent counterparts) accuracies in detecting sentiments in software engineering text.*

Table 11
Contingency matrix of McNemar's test between the accuracies of SentiStrength-SE and the original SentiStrength in *Set-B* dataset

# of comments misclassified by both $\mathcal{T}_a$ and $\mathcal{T}_b$	97	20	# of comments misclassified by $\mathcal{T}_b$ but not by $\mathcal{T}_a$
# of comments misclassified by $\mathcal{T}_a$ but not by $\mathcal{T}_b$	**135**	105	# of comments correctly classified by both $\mathcal{T}_a$ and $\mathcal{T}_b$

Here, $\mathcal{T}_a$ = original SentiStrength and $\mathcal{T}_b$ = SentiStrength-SE

4.3. Evaluating the contribution of domain dictionary

The newly developed software engineering domain dictionary is a major component of SentiStrength-SE. Here, we carry out a quantitative evaluation to verify the contribution of the domain dictionary in detecting sentiments in software engineering texts accurately. Especially, we address the following research question:

RQ3: *Does the domain-specific dictionary in SentiStrength-SEreally contribute to improved sentiment analysis in software engineering text?*

For this particular evaluation, we again use the Group-2 and Group-3 datasets introduced before. We invoke the original SentiStrength for detecting sentiments in issue comments in these datasets. Then, we operate SentiStrength making it use our newly developed domain dictionary and invoke it for sentiment detection in the same issue comments. We use SentiStrength* to refer to the variant of the original SentiStrength that is forced to use our domain dictionary instead of its original one. For each of the three sentimental polarities, we separately compute and compare the precision, recall, and F-score resulting from the tools in each dataset.

4.3.1. Comparison between the original SentiStrength and SentiStrength*

If our domain dictionary actually contributes to improved sentiment analysis in software engineering text, SentiStrength* must perform better than the original SentiStrength. In Table 12, we present the precision (℘), recall (ℜ), and F-score (Ⅎ) obtained in detection of each sentimental polarity. In the table, substantial (i.e., more that 1%) differences are marked in bold.

As seen in Table 12, in every case, SentiStrength* achieves higher F-score than the original SentiStrength. Moreover, SentiStrength* shows *much higher precision* in all cases except for neutral comments in Group-3 dataset. For the neutral comments in Group-3 dataset, the precision of the original SentiStrength is marginally higher by only 0.06%. In all the cases across datasets the precision, recall, and F-score of SentiStrength* is higher or comparable to

Table 12
Comparison of performances of the original **SentiStrength** and **SentiStrength***

Data	Sentiment	Met.	SentiStrength	SentiStrength*
Group-2	Positive	℘	74.48%	87.56%
		ℜ	98.81%	98.28%
		Ⅎ	84.93%	**92.61%**
	Negative	℘	28.22%	**53.19%**
		ℜ	97.66%	97.65%
		Ⅎ	43.78%	**68.87%**
	Neutral	℘	96.83%	**97.94%**
		ℜ	52.42%	**81.85%**
		Ⅎ	68.01%	**89.18%**
Group-3	Positive	℘	31.69%	**40.44%**
		ℜ	**87.79%**	82.01%
		Ⅎ	46.58%	**54.16%**
	Negative	℘	47.61%	**69.10%**
		ℜ	**78.40%**	72.65%
		Ⅎ	59.25%	**70.83%**
	Neutral	℘	91.28%	91.22%
		ℜ	56.16%	**79.54%**
		Ⅎ	69.54%	**84.98%**
Overall average accuracy		℘	61.69%	**73.24%**
		ℜ	78.54%	**85.33%**
		Ⅎ	62.02%	**76.77%**

Here, SentiStrength* is forced to use our domain dictionary instead of its own one.

those of the original SentiStrength. Only in two of the 18 cases (i.e., the recall for positive and negative sentiments in Group-3 dataset), the original SentiStrength's performance is perceived (substantially) better than SentiStrength*. These observations are also reflected in the *overall average accuracies* presented in the bottom three rows of the table. The *overall average accuracies* indicate superior performance of SentiStrength* over the original SentiStrength. Hence, the observed accuracy of SentiStrength* is substantially higher when it is forced to use our new domain dictionary instead of its original one. To determine the statistical significance of our observations, we perform another *McNemar's* test between the results of SentiStrength* and the original SentiStrength. As such, we formulate our null and alternative hypotheses as follows:

Null hypothesis-3 (H_0^3):There is no *significant* difference between the accuracies of the original SentiStrength and SentiStrength*.

Alternative hypothesis-3 (H_a^3): There exist *significant* differences between the accuracies of the original SentiStrength and SentiStrength*.

The contingency matrix for the *McNemar's* test is presented in Table 13. As seen in the contingency matrix, SentiStrength* ($\mathcal{T}_b$) exhibits higher accuracies (compared to the original SentiStrength) as $n_{10} > n_{01}$. The test obtains $p = 2.2 \times 10^{-16}$ where $p < \alpha$. Thus, the test rejects our null hypothesis (H_0^3). Hence the alternative hypothesis (H_a^3) holds true indicating that the difference is statistically significant. Based on these observations and the statistical test, we conclude that our newly created domain dictionary indeed contributes to statistically significant improvements in sentiment analysis. Hence, we answer the research question RQ3 as follows:

> **Ans. to RQ3:** *Our newly created domain dictionary makes statistically significant contributions to the improvement of sentiment analysis in software engineering domain.*

4.4. Our domain dictionary vs. SentiStrength's optimized dictionary

The original SentiStrength has a feature that facilitates optimizing its dictionary for a particular domain (SentiStregth-SE, 0000). We want to verify how our domain dictionary perform in comparison with SentiStrength's dictionary optimized for software engineering text. In particular, we address the following research question:

> **RQ4:** *Can SentiStrength's dictionary optimized for software engineering text perform better than SentiStrength-SE's domain-specific dictionary we created?*

4.4.1. Optimizing SentiStrength's dictionary

SentiStrength's original dictionary can be optimized for a particular domain by feeding it with a set of annotated pieces of texts. To optimize the SentiStrength's dictionary for software engineering domain, we use a dataset consists of Stack Overflow posts/comments related to software engineering. This Stack Overflow posts (SOP) dataset contains total 4,423 comments (Calefato et al., 2017a; Novielli et al., 2018a). Each comment in the SOP dataset is assigned appropriate sentimental polarities (i.e., *positive, negative, neutral*) depending on which ones it expresses. Thus, 35% of posts are labeled with positive sentiment and 27% are labeled with negative sentiment while 38% of the posts are labeled as neutral in sentiment (Calefato et al., 2017a).

Simple annotations with sentimental polarity labels is not enough SentiStrength to be able to use the dataset for optimizing its dictionary. For this purpose, SentiStrength requires a pair of integer sentiment scores $\langle \varrho_c, \mu_c \rangle$ pre-assigned to each comment C where $+1 \leq \varrho_c \leq +5$ and $-5 \leq \mu_c \leq -1$. The interpretation of these score is similar to what is described in Section 2.4. ϱ_c and μ_c respectively represent the positive and negative sentimental scores pre-assigned to the given text C. A given text C labeled to have have positive sentiment, must be assigned a positive sentimental score $\varrho_c > +1$. A higher ϱ_c indicates a higher intensity/strength of the positive emotion. Similarly, a text labeled to have negative sentiment must be assigned a negative sentimental score $\mu_c < -1$. A lower μ_c signifies a higher intensity/strength of the negative emotion expressed in text C. A text labeled as neutral in sentiment, must be assigned sentimental scores $\langle 1, -1 \rangle$.

According to the requirements described above, we derive the sentimental scores for each of the comments in the SOP dataset. For a comment C having positive sentiment, we set $\varrho = +3$. Similarly, we set $\mu = -3$ for a comment expressing negative sentiment. Instead of using extreme values from the domains of ϱ and μ, we choose the ones at the medians. In Table 14, we present examples demonstrating how we assign sentiment scores to the labeled comments in the SOP dataset. This dataset is then fed to the original SentiStrength for optimizing its dictionary for software engineering text. Thus, we produce another variant of the original SentiStrength. We refer to this variant with the optimized dictionary as SentiStrength^O.

4.4.2. Comparison between SentiStrength^O and SentiStrength*

SentiStrength^O and SentiStrength* only differ in their dictionaries. SentiStrength^O uses the optimized dictionary while SentiStrength* uses the dictionary we created for SentiStrength-SE. Thus, comparing between SentiStrength^O and SentiStrength* implies a comparison between SentiStrength's optimized dictionary and SentiStrength-SE's software engineering domain dictionary we created.

We invoke SentiStrength^O for detecting sentiments in issue comments in Group-2 and Group-3 datasets and compute the values of precision ($\wp$), recall ($\Re$), and F-score ($\mathfrak{H}$) in detection of each sentimental polarity. We present the computed metrics values for SentiStrength^O and SentiStrength* side by side in Table 15.

In Table 15, we see that SentiStrength* always achieves higher F-score than SentiStrength^O. Moreover, SentiStrength* achieves *much higher precision* in all cases except for neutral comments in Group-3 dataset. For the neutral comments in Group-3 dataset, the precision of SentiStrength* lower than that of SentiStrength^O marginally by only 0.49%. The recall of SentiStrength* is also substantially higher than or nearly equal to that of SentiStrength^O in 16 of 18 cases. Finally, the overall average accuracies, as presented at the bottom three rows of the table, indicate that the overall precision, recall, and F-score of SentiStrength* are substantially higher than SentiStrength^O.

To determine the statistical significance of our observations, we perform another *McNemar's* test between the results of

Table 13
Contingency matrix for *McNemar's* test between SentiStrength and SentiStrength*

# of comments misclassified by both $\mathcal{T}_a$ and $\mathcal{T}_b$	748	334	# of comments misclassified by $\mathcal{T}_b$ but not by $\mathcal{T}_a$
# of comments misclassified by $\mathcal{T}_a$ but not by $\mathcal{T}_b$	**1,527**	2,955	# of comments correctly classified by both $\mathcal{T}_a$ and $\mathcal{T}_b$

Here, $\mathcal{T}_a$ = original SentiStrength and $\mathcal{T}_b$ = SentiStrength*

Table 14
Examples of assigning sentiment scores to labeled comments

Comment text	Sentiments labeled by human raters	Sentimental scores $\langle \varrho_c, \mu_c \rangle$
@DrabJay: excellent suggestion! Code changed. :-)	Positive	$\langle +3, -1 \rangle$
That really stinks! I was afraid of that...	Negative	$\langle +1, -3 \rangle$
A few but they all seem proprietary	Neutral	$\langle +1, -1 \rangle$

Table 15
Comparison of performances of **SentiStrength^O** and **SentiStrength***

Data	Sentiment	Met.	**SentiStrength^O**	SentiStrength*
Group-2	Positive	℘	74.45%	**87.56%**
		ℜ	**98.68%**	98.28%
		Ⅎ	84.87%	**92.61%**
	Negative	℘	30.12%	**53.19%**
		ℜ	97.66%	97.65%
		Ⅎ	46.04%	**68.87%**
	Neutral	℘	96.69%	**97.94%**
		ℜ	54.29%	**81.85%**
		Ⅎ	69.53%	**89.18%**
Group-3	Positive	℘	26.00%	**40.44%**
		ℜ	**86.90%**	82.01%
		Ⅎ	40.02%	**54.16%**
	Negative	℘	47.13%	**69.10%**
		ℜ	**76.77%**	72.65%
		Ⅎ	58.40%	**70.83%**
	Neutral	℘	91.71%	91.22%
		ℜ	58.02%	**79.54%**
		Ⅎ	71.07%	**84.98%**
Overall average accuracy		℘	61.02%	**73.24%**
		ℜ	78.72%	**85.33%**
		Ⅎ	68.74%	**78.82%**

Here, Note, SentiStrength^O uses the optimized dictionary
SentiStrength* uses our domain dictionary

SentiStrength^O and SentiStrength*. Thus, we formulate our null and alternative hypotheses as follows:

Null hypothesis-4(H_0^4): There is no significant difference between the accuracies of SentiStrength^O and SentiStrength*.

Alternative hypothesis-4(H_a^4): There exist significant differences between the accuracies of SentiStrength^O and SentiStrength*.

The contingency matrix for the *McNemar's* test is presented in Table 16. As seen in the contingency matrix, SentiStrength* ($\mathcal{T}_b$) exhibits higher accuracies (compared to the SentiStrength^O) as $n_{10} > n_{01}$. The test obtains $p = 2.2 \times 10^{-16}$ where $p < \alpha$ and rejects our null hypothesis (H_0^4). Hence the alternative hypothesis (H_a^4) holds true indicating that the difference is statistically significant. The significantly superior accuracies of SentiStrength* implies that the domain dictionary we created for SentiStrength-SE outperforms the original SentiStrength's optimized dictionary. We, therefore, answer the research question RQ4 as follows:

Ans. to RQ4: *The domain dictionary we created for SentiStrength-SEperforms significantly better than the optimized dictionary of the original SentiStrength.*

4.5. Comparison with a large domain-independent dictionary

As mentioned before, domain difficulty is among the major reasons why domain-independent sentiment analysis techniques are found to have performed poorly when operated on in technical texts. This work of ours reveals the same as described in Section 2.5.2. For overcoming the domain difficulties, we have created domain-specific dictionary and heuristics in our SentiStrength-SE. However, compared to the existing domain-independent dictionaries available out there, our domain-specific dictionary is small in size with 167 positively and 310 negatively polarized entries. One might argue that a substantially *large* domain-independent dictionary might not suffer from the domain difficulties we are concerned about and may perform equally, if not better than our relatively small domain-specific dictionary. To verify this possibility, we compare the performances of our domain-specific dictionary with a large domain-independent dictionary. Particularly, we address the following research question:

RQ5: *Can a large domain-independent dictionary perform better than the domain-specific dictionary we created for SentiStrength-SE?*

Table 16
Contingency matrix for *McNemar's* test between SentiStrength^O and SentiStrength*

# of comments misclassified by both $\mathcal{T}_a$ and $\mathcal{T}_b$	942	140	# of comments misclassified by $\mathcal{T}_b$ but not by $\mathcal{T}_a$
# of comments misclassified by $\mathcal{T}_a$ but not by $\mathcal{T}_b$	**1,046**	3,436	# of comments correctly classified by both $\mathcal{T}_a$ and $\mathcal{T}_b$

Here, $\mathcal{T}_a$ = SentiStrength^O and $\mathcal{T}_b$ = SentiStrength*

4.5.1. Choosing a domain independent dictionary for comparison

There are several domain independent dictionaries (e.g., AFINN (Nielsen, 2011), MPQA (Wilson et al., 2009), VADER (Hutto and Gilbert, 2014), SentiWordNet (Baccianella et al., 2010), SentiWords (L. Gatti and Turchi, 2016), and the dictionary of Warriner et al. (2013)) available for sentiment analysis in general. Islam and Zibran (2017a) compared the performances of AFINN (Nielsen, 2011), MPQA (Wilson et al., 2009) and VADER (Hutto and Gilbert, 2014) dictionaries in sentiment analysis of software engineering text. However, all those used dictionaries in the work of Islam and Zibran (2017a) can be considered to have *low* coverage. On the other hand, SentiWordNet (Baccianella et al., 2010), SentiWords (L. Gatti and Turchi, 2016), and the *extended* ANEW (Affective Norms for English Words) dictionary of Warriner et al. (2013) are larger in size and have higher coverage compared to AFINN, MPQA and VADER dictionaries.

Among these three high coverage large dictionaries, we opt for the extended ANEW dictionary of (Warriner et al., 2013), which includes 13,915 English lemmas having 67% reported coverage (L. Gatti and Turchi, 2016). We choose this dictionary for two main reasons: (i) this dictionary has already been used in software engineering studies (Mäntylä et al., 2017; Islam and Zibran, 2018b); (ii) Use of parts-of-speech (POS) as context to determine words' polarities is found to show low accuracy in detecting sentiments in software engineering texts (Islam and Zibran, 2017a). Therefore, we exclude SentiWords and SentiWordNet as these two dictionaries use POS as a context to determine words' polarities.

4.5.2. Range conversion

In the extended ANEW dictionary of Warriner et al. (2013), each word ω is assigned a valence score v_ω, which is a real number between + 1.0 and + 9.0 signifying the sentimental polarity and strength/intensity of the word ω. The sentimental polarity of the word ω, denoted as *sentiment*(ω), is interpreted according to Eq. 7 below.

$$sentiment(\omega) = \begin{cases} Positive, & \text{if } v_\omega > +5.0 \\ Negative, & \text{if } v_\omega < +5.0 \\ Neutral, & \text{otherwise.} \end{cases} \quad (7)$$

In contrast, both the original SentiStrength and our SentiStrength-SE uses integer range [-5, +5] and a different interpretation for the same purpose. To use this extended ANEW dictionary in SentiStrength, we convert the valence score of each word in the extended ANEW dictionary from [+1.0, +9.0] range to [-5, +5] range. In doing that, the fractional value of v_ω is first rounded to its nearest integer $\hat{v}_\omega$. Then, using the conversion scale in Table 17, we convert each integer valence score $\hat{v}_\omega$ in the range [+1, +9] to $\mathcal{V}_\omega$ in the integer range [-5, +5]. For example, if the original valence score of a word rounded to the closest integer is +2, it is converted to -4, according to the mappings shown in Table 17. Such a conversion between ranges does not alter the original valence strength/intensity of the words (Islam and Zibran, 2018b). A similar approach was adopted in a recent work (Islam and Zibran, 2018b) for range conversion of arousal scores.

Table 17
Conversion of valence scores from [+1, +9] to [-5, +5]

Score in [+1, +9]	+1	+2	+3	+4	+5	+6	+7	+8	+9
Score in [-5, +5]	-5	-4	-3	-2	+/- 1	+2	+3	+4	+5

Table 18
Comparison of performances of **SentiStrength^W** and **SentiStrength***

Data	Sentiment	Met.	**SentiStrength^W**	SentiStrength*
Group-2	Positive	℘	50.17%	87.56%
		ℜ	**99.60%**	98.28%
		Ⅎ	66.73%	**92.61%**
	Negative	℘	16.52%	**53.19%**
		ℜ	85.94%	**97.65%**
		Ⅎ	27.71%	**68.87%**
	Neutral	℘	91.11%	**97.94%**
		ℜ	05.86%	**81.85%**
		Ⅎ	11.01%	**89.18%**
Group-3	Positive	℘	10.40%	**40.44%**
		ℜ	**96.79%**	82.01%
		Ⅎ	18.79%	**54.16%**
	Negative	℘	28.33%	**69.10%**
		ℜ	70.29%	**72.65%**
		Ⅎ	40.38%	**70.83%**
	Neutral	℘	75.94%	**91.22%**
		ℜ	08.23%	**79.54%**
		Ⅎ	14.84%	**84.98%**
Overall average accuracy		℘	45.41%	**73.24%**
		ℜ	61.12%	**85.33%**
		Ⅎ	52.10%	**78.82%**

Here, Note, SentiStrength^W uses the extended ANEW dictionary of Warriner et al. (2013)
SentiStrength* uses our domain dictionary (created for SentiStrength-SE)

4.5.3. Comparison between SentiStrength^W and SentiStrength*

We create another variant of the original SentiStrength by replacing its original dictionary with the one created based on the dictionary of Warriner et al., and call this new variant SentiStrength^W. SentiStrength^W is invoked for detecting sentiments in issue comments in Group-2 and Group-3 datasets. Then we compute the precision (℘), recall (ℜ), and F-score (Ⅎ) of SentiStrength^W in detection of each sentimental polarity. The computed metrics values for SentiStrength^W and SentiStrength* are presented side by side in Table 18.

As seen in Table 18, in 16 of the 18 cases SentiStrength* achieves higher precision, recall, and F-score compared to those of SentiStrength^W. SentiStrength*'s recalls for positive sentiments only are slightly lower than those of SentiStrength^W. Notice that, for those same case, the SentiStrength^W's *precision* is substantially lower compared to SentiStrength*. In every case, SentiStrength* maintains a nice balance between precision and recall in detecting sentimental and neutral comments. Such balancing between precisions and recalls results in higher F-scores for SentiStrength* in all cases. The overall accuracies, as presented in the bottom three rows of the table indicates significantly higher precision, recall, and F-score of SentiStrength* compared to SentiStrength^W. To determine the statistical significance of the observed differences in the accuracies, we perform *McNemar's* test between the results of SentiStrength^W and SentiStrength*. For the statistical test, we formulate our null and alternative hypotheses as follows:

Null hypothesis-5 (H_0^5): There is no significant difference between the accuracies of SentiStrength^W and SentiStrength*.

Alternative hypothesis-5(H_a^5): There exist significant differences between the accuracies of SentiStrength^W and SentiStrength*.

The contingency matrix for the *McNemar's* test is presented in Table 19. As seen in the contingency matrix, SentiStrength* ($\mathcal{T}_b$) exhibits higher accuracies (compared to the SentiStrength^W) as $n_{10} > n_{01}$. The test obtains $p = 2.2 \times 10^{-16}$ where $p < \alpha$. Thus, the test rejects our null hypothesis (H_0^5). Hence the alternative hypothesis (H_a^5) holds true indicating that the differences in the accuracies of SentiStrength* and SentiStrength^W are statistically significant. Therefore, we answer the research question RQ5 as follows:

Table 19
Contingency matrix for *McNemar's* test between SentiStrength^W and SentiStrength*

# of comments misclassified by both $\mathcal{T}_a$ and $\mathcal{T}_b$	1,014	68	# of comments misclassified by $\mathcal{T}_b$ but not by $\mathcal{T}_a$
# of comments misclassified by $\mathcal{T}_a$ but not by $\mathcal{T}_b$	**3,270**	1,212	# of comments correctly classified by both $\mathcal{T}_a$ and $\mathcal{T}_b$

Here, $\mathcal{T}_a$ = SentiStrength^W and $\mathcal{T}_b$ = SentiStrength*

> **Ans. to RQ5:** *For sentiment analysis in software engineering text, the domain-specific dictionary we created for SentiStrength-SEperforms significantly better than a large domain-independent dictionary.*

4.5.4. Manual investigation to reveal cause

We conduct an immediate qualitative investigation to reveal why the large dictionary of Warriner et al., having higher coverage, performs worse than our smaller domain-specific dictionary. We identify a set of comments C_m^W from Group-2 dataset, which are misclassified by SentiStrength^W. From the set C_m^W we distinguish another subset C_c^S, which are correctly classified by SentiStrength*. Then we randomly pick 50 comments from the set C_c^S for manual investigation.

From the manual investigation, we find the domain-specific variations in the meaning of words (i.e., the difficulty D_1 revealed in Section 2.5.2) as the main reason for the low accuracies of SentiStrength^W. For example, the following neutral comment is identified to have both negative and positive sentiments by SentiStrength^W.

> "... **crash** for the same reason. Made some local **fixes** here." (Comment ID: 149494)

The above comment is misclassified to have both positive and negative sentiments due to the presence of the words 'Crash' and 'Fix', which are negatively and positively polarized words respectively in the domain-independent ANEW dictionary of Warriner et al. (2013). In software engineering domain, both the words are neutral in sentiments. Due to the same reason, in detection of neutral comments, the performance of the dictionary of Warriner et al. is even worse than both the *optimized* and *default* dictionary of the original SentiStrength.

4.6. Comparison with an alternative domain dictionary

Recall that our domain dictionary for SentiStrength-SE is developed using commit messages only. There is a possibility that a domain dictionary built on text from diverse sources may offer better performance. Thus, to verify this possibility, we create a second domain dictionary using text from diverse sources and compare this new alternative dictionary with the dictionary of SentiStrenght-SE. In particular, we address the following research question:

> **RQ6:** *Can a domain dictionary developed using texts from diverse sources perform substantially better than SentiStrength-SE's domain dictionary developed based on commit messages only?*

4.6.1. Building an alternative domain dictionary

In addition to the 490 thousand commit messages used to develop SentiStrength-SE's dictionary, we obtain 1,600 *Code Review Comments* (CRC) (Ahmed et al., 2017), 1,795 *JIRA Issue Comments* (JIC) (Islam and Zibran, 2018b) and 4,423 Stackoverflow posts

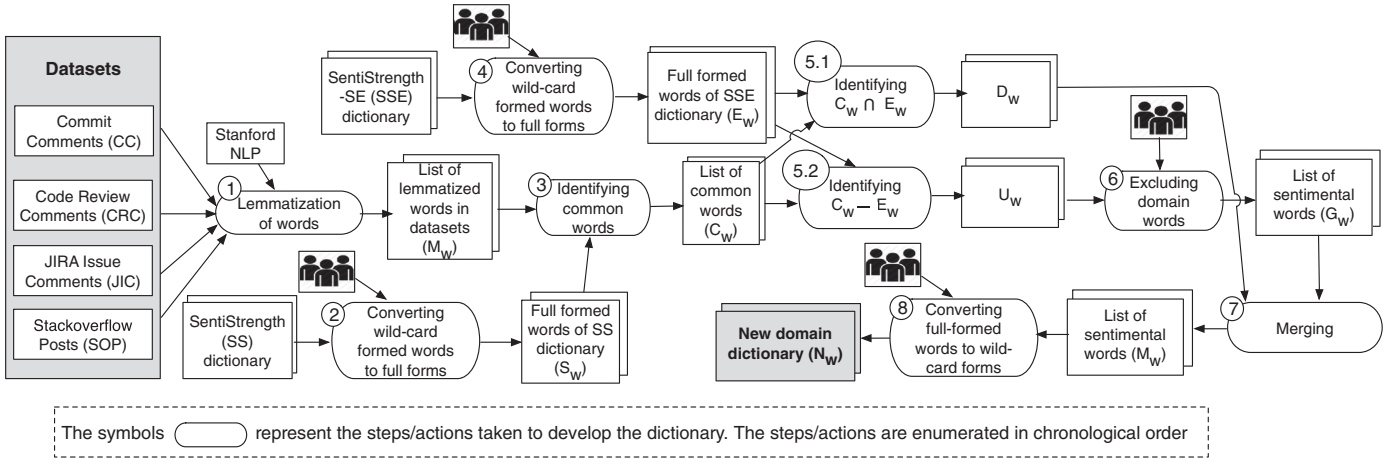

Fig. 3. Procedural steps in developing a new alternative domain dictionary

(SOP) (Calefato et al., 2017a). We use software engineering texts from these diverse datasets to build the alternative domain dictionary.

Fig. 3depicts the steps/actions performed to develop this new dictionary. Here, in the first three steps (i.e., step-1 through step-3), we first produce a set S that includes all distinct words from all the four datasets. Then, we derive another set C_w of 1,198 words that are common in both S and the domain-independent dictionary of the original SentiStrength. These three steps are similar to the first three steps (Fig. 1) in developing the domain dictionary of our SentiStrength-SE . However, here in step-1, we use four datasets instead of commit messages only.

In step-4, we transform the wild-card formed words in SentiStrength-SE's dictionary to their full forms. The set of full-formed words is denoted as E_w. In step-5, we derive a set of words D_w from the set C_w such that D_w contains only those words that are common between C_w and E_w. Mathematically, $D_w = C_w \cap E_w$. We also derive another set U_w, which contains the words that are in C_w but not in E_w. Mathematically, $U_w = C_w - E_w$.

All the words in D_w can safely be considered sentimental as those words are also present in the dictionary of SentiStrength-SE. We need to identify those words in U_w that are neutral in software engineering domain, but could be sentimental in general. Hence, in step-6, we involve three human raters (enumerated as A, B, and C) to independently identify those contextually neutral domain words. These human raters are the same three raters used in the development of the domain dictionary of SentiStrength-SE.

In Table 20, we present sentiment-wise percentage of cases where the human raters disagree pair-wise. We also measure the degree of inter-rater agreement in terms of *Fleiss-κ* (Fleiss, 1971) value. The obtained *Fleiss-κ* value 0.691 signifies substantial agreement among the independent raters. We consider a word as a neutral domain word when two of the three raters identify the word as neutral. Thus, 373 words are identified as neutral domain words, which we exclude from the set U_w resulting in another set of sentimental words G_w. Then, in step-7, by taking a union of the words in the sets G_w and D_w we form another set of words M_w, which contains all the sentimental words. Finally, in step-8, we adjust the words in M_w by reverting them to their wild-card forms (if available) to comply with SentiStrength-SE's dictionary. This new set of words form our new domain dictionary (N_w). At this stage, we also make sure the words, which are manually added in the dictionary of SentiStrength-SE's (see Section 3.1.1), are included in the set of words of the new domain dictionary. This alternative dictionary contains 225 positively and 495 negatively polarized sentimental entries.

Table 20
Inter-rater disagreements in interpretation of sentiments

	Disagreements between human raters		
Sentimental polarity	A, B	B, C	C, A
Positive	11.32%	12.18%	12.22%
Negative	12.25%	11.19%	10.21%
Neutral	12.15%	10.53%	13.42%

4.6.2. Comparison between the new dictionary and SentiStrength-SE's dictionary

We create a variant of SentiStrength-SE by replacing its domain dictionary with the newly created alternative domain dictionary. We call this variant SentiStrength-SE^N. Then we compare the performance of SentiStrength-SE^N against SentiStrength-SE, which actually implies a comparison of SentiStrength-SE's dictionary with the *new* domain dictionary we have created.

We invoke SentiStrength-SE^N for detecting sentiments in issue comments in Group-2 and Group-3 datasets. Then we compute the precision ($\wp$), recall ($\Re$), and F-score ($\exists$) of SentiStrength^W in detection of each sentimental polarity. We present the computed metrics values obtained by SentiStrength-SE^N and SentiStrength-SE side by side in Table 21. As seen in Table 21, SentiStrength-SE^N performs slightly better than SentiStrength-SE in detection of negative sentiments. On the other hand, by observing precision and F-score values, we can say that SentiStrength-SE performs little better in detecting positive and neutral comments, although SentiStrength-SE^N achieves slightly higher recall values in those positive and neutral comments. The overall accuracies, as presented at the bottom three rows of Table 21, indicate that the performances between SentiStrength-SE^N and SentiStrength-SE do not differ substantially. To verify the statistical significance of the observed differences, we perform another *McNemar's* test between the results of SentiStrength-SE^N and SentiStrength-SE. For the statistical test, we formulate our null and alternative hypotheses as follows:

Null hypothesis-6 (H_0^6): There is no significant difference between the accuracies of SentiStrength-SE^N and SentiStrength-SE.

Alternative hypothesis-6(H_a^6): There exist significant differences between the accuracies of SentiStrength-SE^N and SentiStrength-SE.

The contingency matrix for the *McNemar's* test is presented in Table 22. As seen in the contingency matrix, SentiStrength-SE^N ($\mathcal{T}_a$) and SentiStrength-SE ($\mathcal{T}_b$) exhibit almost equal accuracies as $\approxeq$. The test obtains $p = 0.0040$ where $p > \alpha$. Thus, the test fails to reject our null hypothesis (H_0^6). Therefore, we conclude that the performance of the newly created domain dictionary does not significantly differ from that of SentiStrength-SE's domain dictionary. We now formulate the answer to the research question RQ4 as follows:

Table 21
Comparison of performances of **SentiStrength-SE^N** and **SentiStrength-SE**

Data	Sentiment	Met.	**SentiStrength-SE^N**	SentiStrength-SE
Group-2	Positive	℘	87.82%	**88.86%**
		ℜ	98.81%	98.81%
		Ⅎ	92.99%	**93.57%**
	Negative	℘	**53.45%**	53.42%
		ℜ	**97.68%**	97.66%
		Ⅎ	**69.09%**	69.06%
	Neutral	℘	98.13%	**98.14%**
		ℜ	82.57%	**83.00%**
		Ⅎ	89.68%	**89.94%**
Group-3	Positive	℘	39.16%	41.80%
		ℜ	**82.62%**	82.04%
		Ⅎ	53.13%	**55.39%**
	Negative	℘	**70.44%**	68.61%
		ℜ	**72.25%**	71.00%
		Ⅎ	**71.33%**	69.78%
	Neutral	℘	**90.71%**	90.64%
		ℜ	78.94%	**80.05%**
		Ⅎ	84.42%	**85.02%**
Overall average accuracy		℘	73.28%	**73.57%**
		ℜ	**85.47%**	85.43%
		Ⅎ	78.91%	**79.06%**

Here, Note, SentiStrength-SE^N uses the newly created domain dictionary SentiStrength-SE uses its own domain dictionary

Table 22
Contingency matrix for *McNemar's* test between SentiStrength-SE^N and SentiStrength-SE

# of comments misclassified by both $\mathcal{T}_a$ and $\mathcal{T}_b$	1,033	49	# of comments misclassified by $\mathcal{T}_b$ but not by $\mathcal{T}_a$
# of comments misclassified by $\mathcal{T}_a$ but not by $\mathcal{T}_b$	83	4,399	# of comments correctly classified by both $\mathcal{T}_a$ and $\mathcal{T}_b$

Here, $\mathcal{T}_a$ = SentiStrength-SE^N and $\mathcal{T}_b$ = SentiStrength-SE

Ans. to RQ6: *There is no statistically significant difference between the performances of the newly created domain dictionary and the domain dictionary of SentiStrength-SE.*

4.6.3. Manual investigation to determine reasons

The result of the aforementioned comparison appear surprising to us, as we expected the newly created alternative dictionary to perform better than that of SentiStrength-SE. Recall that the newly created dictionary is larger than the domain dictionary of SentiStrength-SE. SentiStrength-SE's dictionary has 167 positively and 310 negatively polarized sentimental words while the newly created one includes 225 positively and 495 negatively polarized words. While large number of entries in a dictionary can be helpful in achieving high recall, they can also misguide the sentiment analysis for a particular domain resulting in low precision. Hence, we manually investigate these possibilities in two phases and identify two reasons why the newly created alternative dictionary failed to outperform that of SentiStrength-SE.

Phase-1 investigation: We randomly select a set of five issue comments for which SentiStrength-SE^N misclassifies their sentiments but SentiStrength-SE correctly classifies. One such comment is as follows:

"I disagree." (Comment ID: 1787887_1)

SentiStrength-SE ^{N} incorrectly identifies the above comment negatively emotional since the word 'disagree' recorded as a negatively polarized word in the newly created dictionary. SentiStrength-SE correctly identifies the comment as neutral as the word is not included in its dictionary. We identifies similar scenarios for all the five randomly picked issue comments.

Cause-1: Some emotional words present in the new domain dictionary also appear in many neutral comments of the ground-truth datasets. Which is why SentiStrength-SE^N ended up misclassifying those neutral comments as sentimental ones. This is a well-known problem of high coverage dictionaries (L. Gatti and Turchi, 2016).

Phase-2 investigation: We randomly pick 20 inherently emotional words from the new domain dictionary, which are not present in the dictionary of SentiStrength-SE. Then, we search for those words in the ground-truth dataset and find five words (among the selected 20 words) (i.e., 'abhor', 'agony', 'appalling', 'crime' and 'delight') do not appear in any comments in the dataset.

Cause-2: This implies, although the new domain dictionary includes more sentimental words compared to the dictionary of SentiStrength-SE, those new sentimental words are unable to create any contribution in sentiment analysis due to their absence in the ground-truth datasets in use.

4.7. Evaluating the contributions of heuristics

In addition to the domain dictionary, SentiStrength-SE also includes a set of heuristics to guide the sentiment detection process towards higher accuracies. The heuristics, particularly designed for software engineering text, are also among the major contributions of this work. Here, we carry out a quantitative analysis to determine to what extent these heuristics contribute in the detection of sentiments in software engineering text. In particular, we address the following research question:

RQ7: *Do the heuristics integrated in SentiStrength-SEreally contribute to improved sentiment analysis in software engineering text?*

We compare the performances of SentiStrength-SE and SentiStrength* to determine the contributions of the heuristics. Recall that SentiStrength* refers to the variant of the original SentiStrength that is forced to use our initial domain dictionary instead of its original one. Thus, SentiStrength-SE and SentiStrength* use the same domain dictionary and the only difference between them is the set of heuristics that are included in SentiStrength-SE. Hence, the heuristics are liable for any differences between the performances of SentiStrength-SE and SentiStrength*.

We present the performances of SentiStrength-SE and SentiStrength* in Table 23. For determining the effects of heuristics included in SentiStrength-SE, let us compare the right-most two columns in Table 23. We observe that the precision, recall, and F-score achieved by our SentiStrength-SE are consistently higher than those of SentiStrength* in most cases. In a few cases for the Group-3 dataset, SentiStrength-SE's accuracy is nearly equal to those of SentiStrength*. The overall average accuracies, as presented at the bottom of Table 23, also indicate the superiority of our SentiStrength-SE over SentiStrength*, which implies that the heuristics incorporated in SentiStrength-SE really contribute to higher accuracy in the detection of sentiments in software engineering text.

However, as seen in Table 23, although the accuracy of SentiStrength-SE is higher compared to SentiStrength*, they do not differ by a large margin. Thus, it appears that the contributions of heuristics, in this particular case, are not substantial and unlikely to be statistically significant. To verify our observations, we perform a *McNemar's* test between the results of SentiStrength* and SentiStrength-SE. For the test, we formulate our null and alternative hypotheses as follows:

Null hypothesis-5(H_0^7): There is no significant difference between the accuracies of SentiStrength-SE and SentiStrength*.

Alternative hypothesis-5(H_a^7): There exist significant differences between the accuracies of SentiStrength-SE and SentiStrength*.

Table 23
Contributions of heuristics in SentiStrength-SE

Data	Sentiment	Met.	SentiStrength-SE	SentiStrength*
Group-2	Positive	℘	**88.86%**	87.56%
		ℜ	**98.81%**	98.28%
		Ⅎ	**93.57%**	92.61%
	Negative	℘	**53.42%**	53.19%
		ℜ	**97.66%**	97.65%
		Ⅎ	**69.06%**	68.87%
	Neutral	℘	**98.14%**	97.94%
		ℜ	**83.00%**	81.85%
		Ⅎ	**89.94%**	89.18%
Group-3	Positive	℘	**41.80%**	40.44%
		ℜ	**82.04%**	82.01%
		Ⅎ	**55.39%**	54.16%
	Negative	℘	68.61%	**69.10%**
		ℜ	71.00%	**72.65%**
		Ⅎ	69.78%	**70.83%**
	Neutral	℘	90.64%	**91.22%**
		ℜ	**80.05%**	79.54%
		Ⅎ	**85.02%**	84.98%
Overall average accuracy		℘	**73.58%**	73.24%
		ℜ	**85.43%**	85.33%
		Ⅎ	**79.06%**	78.82%

Here, SentiStrength is forced to use our domain dictionary instead of its own one.

The contingency matrix for the *McNemar's* test is presented in Table 24. As seen in the contingency matrix, SentiStrengthSE ($\mathcal{T}_a$) and SentiStrength* ($\mathcal{T}_b$) exhibit almost equal accuracies as $n_{10} \approxeq n_{01}$. The test obtains $p = 0.874$ where $p > \alpha$. Thus, the test fails to reject our null hypothesis (H_0^7). Therefore, we conclude that the contribution of the heuristics in the tool is not significant in this particular case.

Based on our observations from the quantitative analysis and the statistical test, we now formulate the answer to the research question RQ5 as follows:

> **Ans. to RQ7:** *Although the set of heuristics integrated in SentiStrength-SEcontribute towards improved sentiment analysis in software engineering text, the perceived improvement is not statistically significant for the given datasets.*

Recall that, from the exploratory study (Section 2) using the Group-1 portion of the "Gold Standard" dataset, we found that the majority of the misclassifications of sentimental polarities are due to the limitations (difficulties D_1, D_3, D_6) of the dictionary in use (Table 3). Hence, the majority of misclassifications are to be corrected by using a domain dictionary, *leaving a relatively narrow scope for further contributions from the heuristics*, at least for this "Gold Standard" dataset. Our manual investigation of the datasets used in this study confirms the existence of *very few issue comments within operational scope of the heuristics*.

4.7.1. Further manual investigation

Although SentiStrength-SE is found to have performed better in most cases, as seen in Table 23, in four cases for the Group-3 dataset SentiStrength-SE's accuracy is *marginally* lower than SentiStrength*. An immediate qualitative investigation reveals two reasons for this, which we discuss now.

First, our parameter setting to identify negations of sentimental words falls short in capturing negations in some cases. For example, in the following issue comment, the negation word "Don't" preceding the word 'Know' neutralizes the negatively polarized word 'Hell' due to the negation configuration parameter set to five in SentiStrength-SE.

> "I **don't** know how the **hell** my diff program decide to add seemingly random CR chars, but i've removed them now"
> (Comment ID: 306519_2)

A lower negation parameter could work better for this particular issue comment, but might perform worse for negations in others. Other possible solutions are discussed in Section 4.9.

Second, we also found instances of incorrect annotations for negative sentiments for some issue comments in the Group-3 dataset, which caused the accuracy of SentiStrength-SE appear to go down. Consider the following two issue comments.

> "Inserting timestamps automagically would be bad because it would limit a whole swath of use cases"
> (Comment ID: 1462480_2)

> "If that would be the case, this would be bad design"
> (Comment ID: 748115_2)

In the above two issue comments, the author stated merely the possibilities of negative scenarios that hadn't happened yet. These comments do not convey negative sentiments. But the human raters annotated with negative sentiments possibly due considering the use of negatively polarized word 'Bad' in the sentences.

These observations inspire us to conduct a deeper qualitative investigation of the success scenarios and especially the failure cases of SentiStrength-SE mainly to explore opportunities for further improvements to the tool. Hence, a qualitative evaluation of SentiStrength-SE is presented in the following section.

Table 24
Contingency matrix for *McNemar's* test between SentiStrength-SE and SentiStrength*

# of comments misclassified by both $\mathcal{T}_a$ and $\mathcal{T}_b$	993	78	# of comments misclassified by $\mathcal{T}_b$ but not by $\mathcal{T}_a$
# of comments misclassified by $\mathcal{T}_a$ but not by $\mathcal{T}_b$	**81**	4,433	# of comments correctly classified by both $\mathcal{T}_a$ and $\mathcal{T}_b$

Here, $\mathcal{T}_a$ = SentiStrength-SE and $\mathcal{T}_b$ = SentiStrength*

4.8. Qualitative evaluation of SentiStrength-SE

Although from the comparative evaluations we found our SentiStrength-SE superior to the all selected tools, SentiStrength-SE is not a foolproof sentiment analysis tool. Indeed, 100% accuracy cannot be a pragmatic expectation. Nevertheless, we carry out another qualitative evaluation of SentiStrength-SE with two objectives: first, to confirm that the achieved accuracy found in the comparative evaluations did not occur by chance, and second, to identify failure scenarios and scopes for further improvements.

We first randomly pick 150 issue comments (50 positive, 50 negative, and 50 sentimentally neutral) from the Group-2 and Group-3 of the "Gold Standard" dataset for which SentiStrength-SE correctly detected the sentimental polarities. From our manual verification over these 150 issue comments, we are convinced that the design decisions, heuristics, and parameter configuration adopted in SentiStrength-SE have positive impacts on the accurate detection of sentimental polarities.

Next, we randomly choose another 150 issue comments (50 positive, 50 negative, and 50 sentimentally neutral) for which SentiStrength-SE failed to correctly detect the sentimental polarities. Upon manual investigation of those 150 issue comments, we find a number of reasons for the inaccuracies, a few of which are within the scope of the design decisions applied to SentiStrength-SE, and the rest falls beyond, which we discuss in Section 4.9. One of the reasons for failure is missing sentimental terms in our newly created domain dictionary. For example, SentiStrength-SE incorrectly identified the following comment as neutral in sentiment by misinterpreting the sentimental word 'Stuck' as a neutral sentimental word, since the word was not included in the dictionary, which we add to the dictionary of

SentiStrength-SE's release.

"For the first part, I got **stuck** on two points" (Comment ID: 1610758_3)

Some other cases we have found inconsistencies in human rating of sentiments in issue comments, which are liable for inaccuracy in SentiStrength-SE. For example the following comment is rated as neutral in sentiment by human raters, although that contains the positive sentimental term 'Thanks' along with the exclamatory sign '!'.

"And many **thanks** to you Oliver for applying this so quickly **!** " (Comment ID: 577184_1)

Our investigation reveals that 200 issue comments are wrongly interpreted in Group-3 by human raters that cause low accuracy in SentiStrength-SE for detecting positive sentiment, which is aligned with our earlier findings of such wrong interpretation of sentiments.

Although the additional preprocessing phase of SentiStrength-SE filters out unwanted content such as source code, URL, numeric values from the input texts, we found several instances where such contents escaped the filtering technique and misguided the tool.

In a few cases, we found that our heuristics to identify proper nouns fell short for not taking into account probable cases. For example, SentiStrength-SE incorrectly computed negative sentiment in the following issue comment. As seen in the following comment a developer thanked his colleague name 'Harsh'.

"Thanks **Harsh ,** the patch looks good ... Since this is a new API, we are not sure if want to change it. Let's leave it as-is for the moment." (Comment ID: 899420)

For failing to identify 'Harsh' as a proper noun, SentiStrength-SE considered the word sentimentally negative and erroneously detects negative sentiment in the message. Our immediate future plan includes further extension to our heuristics for locating proper nouns in text.

4.9. Threats to validity

In this section, we discuss the threats to the validity of the empirical evaluation of SentiStrength-SE and our efforts in mitigating them.

4.9.1. Construct validity and internal validity

Threats to construct validity relate to the suitability of the evaluation metrics. We use three metrics: *precision, recall* and *F-score* to evaluate the classification performances of SentiStrength-SE and other tools. All the three metrics have been commonly used for similar purposes in software engineering studies (Ahmed et al., 2017; Blaz and Becker, 2016; Calefato et al., 2017a). Only quantitative analysis may not portray the whole picture, which is why we have performed both quantitative and qualitative evaluation of SentiStrength-SE.

The accuracies in the computations of the metrics are subject to the correctness in the manual annotation of the issue comments with sentimental polarities. Hence, we have manually checked the annotations of issue comments in the "Gold Standard" dataset. We identified around 200 issue comments that are incorrectly labelled with wrong sentimental polarities. Nevertheless, we did not exclude those misclassified issue comments because they equally affect all the tools without favoring one over another.

To compare the performance of SentiStrength-SE against other tools (e.g., SentiStrength, NLTK, and StanfordNLP), we have used their default settings. Different settings of those tools might provide different results, however we adhered to their default settings due to their uses in earlier software engineering studies (Guzman et al., 2014; Guzman and Bruegge, 2013; Pletea et al., 2014; Tourani et al., 2014; Ortu et al., 2015; Calefato and Lanubile, 2016; Chowdhury and Hindle, 2016; Garcia et al., 2013; Jongeling et al., 2015; Rahman et al., 2015; Rousinopoulos et al., 2014).

While optimizing SentiStrength's default dictionary for software engineering domain, we assigned three constant values +3, -3 and ± 1 to positive, negative and neutral comments, respectively. One may question the reasons for choosing those particular values instead of other values in the domains. For example, we could have used +2, +4 or +5 instead of +3 and -2,-4 or -5 instead of -3. Recall that those integer values not only indicate polarities of sentiments but also their intensities/strength (SentiStregth-SE, 0000). In the computation of precision, recall and F-score, only the sentimental polarities are considered. Hence any value from +2 to +5 for positively polarized comments and any value from -2 to -5 for negatively polarized comments could be used in the process of optimization. We simply picked the values in the median, instead of choosing extreme ones at the domain boundaries.

We changed the range of valence scores of the words from [+1, +9] to [-5, +5] in the ANEW dictionary of Warriner et al. (2013) to compare its performance against the domain dictionary of SentiStrength-SE. One might argue that the range conversion might have altered the original sentimental polarities of some words. We have considered this possibility and carefully designed the conversion scheme to minimize such possibilities. A random sanity check after the range conversion indicates absence of any such occurrence.

4.9.2. External validity

The use of only one benchmark dataset (i.e., the "Gold Standard" dataset) can be considered a limitation of the empirical evaluation of our SentiStrength-SE. The outcome of the work could be more generalizable if more than one benchmark datasets could be used. At the time of first release of SentiStrength-SE, this "Gold Standard" dataset has been the only publicly available dataset especially crafted for the software engineering domain (Ortu et al., 2016b; Islam and Zibran, 2017b). A few newer datasets are available, but those are either not software engineering domain specific or they are even more specific to a narrower context (e.g., code review, product review). The issue comments in the benchmark dataset are collected from open-source systems and thus one may question whether or not the tools including ours will perform differently if applied on datasets drawn from industrial/proprietary projects. Producing a large dataset with human annotations is a tedious and time consuming task. We are working towards creating a second benchmark dataset for sentiment analysis in software engineering text. Once completed, we will release the dataset for the community.

Although there are diverse sources of textual content produced at different stages of software development and maintenance, the benchmark dataset we used contains only JIRA issue comments. Hence, one may argue that the results of the empirical comparison of tools might substantially vary if a dataset with a different type of text is used. Recall that the dictionary of our SentiStrength-SE tool is created based on commit comments. Thus, its superior performances on issue comments give us confidence that the tool will also perform well on other types of textual content.

4.9.3. Reliability

The methodology of this study including the procedure for data collection and analysis is documented in this paper. The "Gold Standard" dataset (Ortu et al., 2016b) and all the tools (i.e., SentiStrength-SE (SentiStregth-SE, 0000), SentiStrength (Thelwall et al., 2012), NLTK (NLTK, 0000) and Stanford NLP (StanfordCoreNLP, last access: June 2018)) are available freely available online. Therefore, it should be possible to replicate the empirical evaluation of our tool.

5. Limitations of SentiStrength-SE and future possibilities

In this section, we discuss the limitations in the design and implementation of SentiStrength-SE as well as some directions for further improvements to our tool. In the development of SentiStrength-SE, we have addressed the difficulties identified

from the exploratory study described in Section 2. Still there are scopes for further improvements, as we also found from the qualitative evaluation of the tool. For example, we have observed in Section 4.8 that missing of sentimental words can mislead the SentiStrength-SE. We have created an alternative new domain dictionary for SentiStrength-SE using texts from diverse sources. Surprisingly, this newly created domain dictionary do not offer significant performance improvements. We plan to further extend our domain dictionary by integrating with different dictionary building approaches (Blaz and Becker, 2016; Dragut et al., 2010; Passaro et al., 2015).

Our adopted approach for domain dictionary creation is unique from other attempted approaches (Blaz and Becker, 2016; Dragut et al., 2010; Passaro et al., 2015). We have deliberately chosen this approach for two reasons. First, we wanted to introduce a new approach, and second, it was not possible to adopt existing approached due to limitation of resources such as sentiment-annotated texts in software engineering (Ortu et al., 2016b). Through the empirical evaluations, we have shown that our created domain dictionary is effective for sentiment analysis in software engineering. Moreover, in the process of development of the dictionary the identification of domain terms by three human raters might cause a subjectivity bias. However, we have measured the inter-rater agreements, and found reasonable agreements, which minimizes the threat substantially.

In the creation of our new domain dictionary, we used graduate students as the human raters, rather than expert software developers from industry. However, all the participants have at least three years of software development experience, which mitigates this threat. Moreover, it is reported that only minor differences exist between the performances of graduate students and professional software developers especially at small tasks involving simple judgements (Host et al., 2000).

The use of only three human raters may be argued as a small number of participants. However, two to three raters have been common practice in successful software engineering studies (Ahmed et al., 2017; Blaz and Becker, 2016; Calefato et al., 2017a; Panichella et al., 2015). Moreover, through the empirical evaluations (both quantitative and qualitative), we have shown that our created domain dictionary is effective for sentiment analysis in software engineering text.

Several methods have been proposed and discussed to identify the scope of the negations of polarized words in sentiment analysis (Prollochs et al., 2016; Asmi and Ishaya, 2012), which can be applied to improve the performance of our negation handling approach. Many sophisticated approaches to identify the negation's scope include machine learning techniques (Prollochs et al., 2016; Morante et al., 2008), complex rules (Jia et al., 2009), and identifying negated words using semantics of phrases (Choi and Cardie, 2008). However, many existing sentiment analysis approaches have relatively simple methods to identify scope of negation (Panga et al., 2002; Kennedy and Inkpen, 2006). Interestingly, performance of a negation detection method can be improved by domain adaptation (Wu et al., 2014). In future, we will evaluate all the mentioned methods by applying in software engineering contexts to identify the best method to detect scope of negation.

Although our approach for filtering out code snippets may not correctly locate all code portions, but the filtering indeed minimizes them. Indeed, isolating inline source code from plain text content is a challenging task, especially when the text can have code written in diverse undeclared programming language. Such a code separation problem can be a separate research topic and limited scope attempts are made in the past (Bacchelli et al., 2011). We also plan to invest efforts along this direction to further improve SentiStrength-SE.

At this stage, we did not address the difficulties D_{10}, D_{11}, and D_{12}, which are included in our future plan. The detection of irony, sarcasm, and subtle emotions hidden in text is indeed a challenging research topic in NLP and not only related to software engineering texts. Even human interpretations of sentiments in text often disagree as such we also found in the "Gold Standard" dataset. Combining the dictionary-based lexical method with machine learning (Reyes et al., 2012) and other specialized techniques (Balahur et al., 2011) can lead to potential means to address these difficulties. We also plan to add to SentiStrength-SE the capability to identify interrogative sentences correctly mitigate the difficulty D_{10}.

6. Related work

To the best of our knowledge, the qualitative study (Section 2), is the first study that analyzes *public benchmark dataset* to expose the challenges to sentiment analysis in software engineering. And, we have developed the first sentiment analysis tool, SentiStrength-SE, crafted especially for software engineering domain, which we expect to produce superior performance in other technical domains as well.

Aside from our tool, there are only four prominent tools/toolkits namely, SentiStrength (Thelwall et al., 2012), Stanford NLP (StanfordCoreNLP, last access: June 2018), NLTK (NLTK, 0000), and Alchemy (AlchemyLanguage, 0000), which facilitate automatic sentiment analysis in plain texts. The first three of these tools have been used for sentiment analysis in software engineering domain, while SentiStrength is used most frequently in the studies as presented in Table 25. We categorize those studies for better understanding of the uses of those tools and the contributions of those studies in software engineering domain. Those tools, which are previously used in software engineering area, but *not for sentiment analysis*, are excluded from the table. Notably, none of the studies used any domain specific tool to detect sentiments.

Alchemy (AlchemyLanguage, 0000) is a commercial toolkit that offers limited sentiment analysis as a service through its published APIs. According to the study of Jongeling et al. (2017) the performance of Alchemy is lower than SentiStrength (Thelwall et al., 2012) and NLTK (NLTK, 0000). NLTK and Stanford

Table 25
Uses of tools for *sentiment analysis* and their contributions in software engineering

Tools	Type of work	Uses in software engineering research
SentiStrength (Thelwall et al., 2012)	Analyzing sentiments in software engineering (SE)	Guzman et al. (2014); Tourani et al. (2014); Islam and Zibran (2016a,b); Chowdhury and Hindle (2016); Novielli et al. (2015); Guzman (2013); Ortu et al. (2016a)
	Applications of sentiments in SE	Guzman and Bruegge (2013); Ortu et al. (2015); Calefato and Lanubile (2016); Garcia et al. (2013); Tourani and Adams (2016); Guzman and Maalej (2014); Jiarpakdee et al. (2016)
	Benchmarking study	Jongeling et al. (2015, 2017)
NLTK (NLTK, 0000)	Analyzing sentiments in SE	Pletea et al. (2014); Rousinopoulos et al. (2014)
	Benchmarking study	Jongeling et al. (2015, 2017)
Stanford NLP (StanfordCoreNLP, last access: June 2018)	Applications of sentiments in SE	Rahman et al. (2015)
	Benchmarking study	Jongeling et al. (2015, 2017)

NLP (StanfordCoreNLP, last access: June 2018) are general purpose natural language processing (NLP) library/toolkit, which expect the user to have some NLP background and to write scripting code for carrying out sentiment analysis in plain text. In contrast, SentiStrength is a dedicated tool that applied a lexical approach for automated sentiment analysis and is ready to operate without needing to write any scripting code (for natural language processing). Perhaps, these are among the reasons why, in software engineering community, SentiStrength has gained popularity over the alternatives. The same reasons also made us choose this particular tool as the basis of our work. Our SentiStrength-SE reuses the lexical approach of the original SentiStrength and is also ready to be used off the shelf.

All of the aforementioned four tools (i.e., SentiStrength (Thelwall et al., 2012), Stanford NLP (StanfordCoreNLP, last access: June 2018), NLTK (NLTK, 0000), and Alchemy (AlchemyLanguage, 0000)) are developed and trained to operate on non-technical texts drawn from social interactions, web pages, and they do not perform well enough when operated in a technical domain such as software engineering. Domain-specific (e.g., software engineering) technical uses of inherently emotional words seriously mislead the sentiment analyses of those tools (Pletea et al., 2014; Tourani et al., 2014; Jongeling et al., 2015; Novielli et al., 2015) and limit their applicability in software engineering area. We have addressed this issue by developing the first software engineering domain-specific dictionary included in our tool SentiStrength-SE. Along this direction Mäntylä et al. (2016) developed a dictionary to capture *emotional arousal* in the software engineering texts.

Apart from creating domain dictionary, a variety of machine learning (ML) techniques such as, Naive Bayes classifier (NB), Support Vector Machine (SVM) (Panga et al., 2002), and Logistic Regression (LR) (Choudhury et al., 2012) have been explored in an attempt to minimize the domain difficulty. However, the performances of all these three classifiers are reported lower when operated on domain-specific texts (Muhammad et al., 2013). Nonetheless, recently Murgia et al. (2017) applied several ML techniques (e.g., NB, SVM) to identify emotions *love, joy* and *sadness* only in contrast to our tool SentiStrength-SE that can differentiate *positivity, negativity* and *neutrality* of software engineering texts. Again, Panichella et al. (2015) used NB classifier to detect sentiments in software users' reviews. However, the accuracy of their classifier was not reported. Those two tools are not publicly available to compare against our tool SentiStrength-SE. Moreover, we avoided to apply ML technique to implement SentiStrength-SE due to the limitations of ML for sentiment analysis that include its difficulty to integrate into a classifier and learned models often have poor adaptability between different text genres or domains as they often rely on domain specific features found in their training data (Muhammad et al., 2013).

Blaz and Becker (2016) proposed three almost equally performing methods, a *Dictionary Method (DM)*, a *Template Method (TM)* and a *Hybrid Method (HM)* for sentiment analysis in "Brazilian Portuguese" texts in IT (Information Technology) job submission tickets. The DM is a pure lexical approach similar to that of our SentiStrength-SE. Although their techniques might be suitable for formally structured texts, those may not perform well in dealing with informal texts that are frequently used in software engineering artifacts such as commit comments. In contrast, from the empirical evaluation over commit comments, our SentiStrength-SE is found to have high accuracy in detecting sentiments in those informal software engineering texts. The proposed methods of Blaz and Becker (2016) are developed and evaluated against text written in "Brazilian Portuguese" language instead of English. Thus, their approach and reported results are not directly comparable to ours.

Similar to the qualitative study included in our work, Novielli et al. (2015) also conducted a relatively brief study of the challenges against sentiment analysis in "social programmer ecosystem". They also used SentiStrength for the detection of emotional polarities and reported only domain difficulty as a key challenge. In their work, they manually studied only 100 questions and their follow-up comments as well as 100 answers and their follow-up discussions obtained from Stack Exchange Data Dump (Stack Exchange Data Dump, 0000). In contrast, based on a deeper analysis over a publicly available benchmark dataset, our study exposes 12 difficulties including the domain dependency. In addition, we address a portion of those difficulties and develop a domain-specific tool for improved sentiment analysis in software engineering text.

Our SentiStrength-SE is the *first* software engineering domain specific sentiment analysis tool. Soon after the release of SentiStrength-SE, four domain-specific tools/toolkits (i.e., Senti4SD (Calefato et al., 2017a), SentiCR (Ahmed et al., 2017), EmoTxt (Calefato et al., 2017b), and SentiSW (Ding et al., 2018)) have appeared over the last few months. Similar to our SentiStrength-SE, Senti4SD is also a software engineering domain specific sentiment analysis tool. Senti4SD applies machine learning based on lexicon and keyword based features for detecting sentiments. SentiSW also applies machine learning techniques to detect sentiments at *entity-levels*. EmoTxt (Calefato et al., 2017b) is an open-source toolkit for detecting six basic emotions (i.e., love, joy, anger, sadness, fear, and surprise) from *technical text*. The authors of SentiCR declared the scope of this tool limited to code review comments only. Again, the applicability of SentiSW is limited to only JIRA issue comments. The reported scope of EmoTxt is technical domain, which is wider than software engineering domain. On the contrary, the scopes of SentiCR and SentiSW are limited to narrower domains of code review comments and JIRA issue comments, respectively.

Two separate studies were conducted to compare the performances of those domain specific sentiment analysis tools. In the first study, Islam and Zibran (2018a) compared the performances of SentiStrength-SE, Senti4SD and EmoTxt and found no convincing winner among the tools for sentiment analysis in software engineering. In the later study, Novielli et al. (2018b) compared the performances of SentiStrength-SE, Senti4SD and SentiCR and found the unsupervised approach of SentiStrength-SE had provided comparable performance to that of supervised techniques (i.e., Senti4SD and SentiCR). SentiSW was not available at the time of conducting the two comparative studies. However, developers of SentiSW compared its performance against SentiStrength-SE and found their tool's superiority over SentiStrength-SE in detecting sentiments expressed in JIRA issue comments. While all those studies compared the performances of domain specific tools, for the first time, Jongeling et al. (2015, 2017) compared the performances of domain independent tools SentiStrength (Thelwall et al., 2012), NLTK (NLTK, 0000), Stanford NLP (StanfordCoreNLP, last access: June 2018) and Alchemy (AlchemyLanguage, 0000) to measure their applicability in software engineering domain.

We do not claim SentiStrength-SE to be the best tool among all the few tools available for sentiment analysis in software engineering text. Instead, by developing and evaluating the *domain-specific* SentiStrength-SE, we demonstrate that, for sentiment analysis in software engineering text, a *domain-specific* technique performs significantly better compared to its *domain-independent* counterparts. As mentioned before, our SentiStrength-SE is the first domain-specific tool for sentiment analysis in software engineering. Other researches might have taken motivations from our work (Islam and Zibran, 2017b) in attempting domain-specific solutions resulting in several domain-specific tools discussed above.

7. Conclusion

In this paper, we have first presented an in-depth qualitative study to identify the difficulties in automated sentiment analysis in software engineering texts. Among the difficulties, the challenges due to domain dependency are found the most dominant. To address mainly the

domain difficulty, we have developed a domain-specific dictionary especially designed for sentiment analysis in software engineering text.

We also develop a number of heuristics to address some of the other identified difficulties. Our new domain dictionary and the heuristics are integrated in SentiStrength-SE, a tool we have developed for improved sentiment analysis in textual contents in a technical domain, especially in software engineering. Our tool reuses the lexical approach of SentiStrength (Thelwall et al., 2012), which, in software engineering, is the most widely adopted sentiment analysis technique. Our SentiStrength-SE is the first domain-specific sentiment analysis tool especially designed for software engineering text.

Over a large dataset (i.e., Group-2 and Group-3) consisting of 5,600 issue comments, we carry out quantitative comparisons of our *domain-specific* SentiStrength-SE with the three most popular *domain independent* tools/toolkits (i.e., NLTK (NLTK, 0000), Stanford NLP (Socher et al., 2013b), and the original SentiStrength (Thelwall et al., 2012). The empirical comparisons suggest that our domain-specific SentiStrength-SE is significantly superior to its *domain independent* counterparts in detecting emotions in software engineering textual contents.

Using both quantitative and qualitative evaluations, we also separately verify the effectiveness of the design decisions including the domain dictionary and heuristics we have included in our *domain-specific* SentiStrength-SE. From the evaluations, we found that our newly created domain dictionary makes statistically significant contributions to improved sentiment analysis in software engineering text. However, the heuristics we developed to minimize the issues beyond the domain difficulties are found not to have substantial impacts on sentiment analysis of the chosen datasets. The non-substantial impact of the heuristics further validates that the improvements in the accuracies of SentiStrength-SE are attributed to its being *domain-specific*. Thus, we demonstrate that, for sentiment analysis in software engineering text, a domain-specific technique performs substantially better than domain independent techniques.

In future, we plan to further verify these findings by extending SentiStrength-SE and operating it on industrial/proprietary datasets. Both from the exploratory study and qualitative evaluation of our sentiment analysis tool, we have also identified scopes for further improvements of the tool, which remain within our future research plans. Using SentiStrength-SE and its future releases, we also plan to conduct large scale studies of emotional variations and their impacts using both public and proprietary datasets in software engineering domain. The current release of our SentiStrength-SE is made freely available (SentiStregth-SE, 0000) for public use.

References

AlchemyLanguage: Natural language processing for advanced text analysis. http://www.alchemyapi.com/products/alchemylanguage/sentiment-analysis.

Gold Standard Dataset Labeled with Manually Annotated Emotions. http://ansymore.uantwerpen.be/system/files/uploads/artefacts/alessandro/MSR16/archive3.zip.

Jazzy- The Java Open Source Spell Checker. http://jazzy.sourceforge.net.

Stack Exchange Data Dump. https://archive.org/details/stackexchange.

Ahmed, T., Bosu, A., Iqbal, A., Rahimi, S., 2017. Senticr: a customized sentiment analysis tool for code review interactions. Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering. pp. 106–111.

Asmi, A., Ishaya, T., 2012. Negation identification and calculation in sentiment analysis. Proceedings of the Second International Conference on Advances in Information Mining and Management. pp. 1–7.

Bacchelli, A., Cleve, A., Lanza, M., Mocci, A., 2011. Extracting structured data from natural language documents with island parsing. Proceeding of the International Conference on Automated Software Engineering. pp. 476–479.

Baccianella, S., Esuli, A., Sebastiani, F., 2010. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. Proceeding of the International Conference on Language Resources and Evaluation. pp. 2200–2204.

Balahur, A., Hermida, J., Montoyo, A., 2011. Detecting implicit expressions of sentiment in text based on commonsense knowledge. Proceedings of the Workshop on Computational Approaches to Subjectivity and Sentiment Analysis. pp. 53–60.

Bettenburg, N., Adams, B., Hassan, A., 2011. A lightweight approach to uncover technical information in unstructured data. Proceedings of the International Conference of Program Comprehension. pp. 185–188.

Blaz, C., Becker, K., 2016. Sentiment analysis in tickets for it support. Proceedings of the International Conference on Mining Software Repositories. pp. 235–246.

Calefato, F., Lanubile, F., 2016. Affective trust as a predictor of successful collaboration in distributed software projects. Proceedings of the International Workshop on Emotion Awareness in Software Engineering. pp. 3–5.

Calefato, F., Lanubile, F., Maiorano, F., Novielli, N., 2017. Sentiment polarity detection for software development. Emp. Softw. Eng. 352–1382.

Calefato, F., Lanubile, F., Novielli, N., 2017. EmoTxt: A toolkit for emotion recognition from text. Proceedings of the Affective Computing and Intelligent Interaction.

Choi, Y., Cardie, C., 2008. Learning with compositional semantics as structural inference for subsentential sentiment analysis. Proceedings of the Conference on Empirical Methods in Natural Language Processing. pp. 793–801.

Choudhury, M., Counts, S., 2013. Understanding affect in the workplace via social media. Proceedings of the Computer supported cooperative work. pp. 303–316.

Choudhury, M., Gamon, M., Counts, S., 2012. Happy, nervous or surprised? Classification of human affective states in social media. Proceedings of the International AAAI Conference on Weblogs and Social Media. pp. 435–438.

Chowdhury, S., Hindle, A., 2016. Characterizing energy-aware software projects: Are they different? Proceedings of the International Conference on Mining Software Repositories. pp. 508–511.

Destefanis, G., Ortu, M., Counsell, S., Marchesi, M., Tonelli, R., 2015. Software development: do good manners matter? PeerJ PrePrints 1–17.

Dewan, P., 2015. Towards emotion-based collaborative software engineering. Proceedings of the International Workshop on Cooperative and Human Aspects of Software Engineering. pp. 109–112.

Dietterich, T., 1998. Approximate statistical tests for comparing supervised classification learning algorithms. J. Neur. Comput. 10 (7), 1895–1923.

Ding, J., Sun, H., Wang, X., Liu, X., 2018. Entity-level sentiment analysis of issue comments. Proceeding of the Third International Workshop on Emotion Awareness in Software Engineering.

Dragut, E., Yu, C., Sistla, P., Meng, W., 2010. Construction of a sentimental word dictionary. Proceedings of the International Conference on Information and Knowledge Management. pp. 1761–1764.

Fleiss, J., 1971. Measuring nominal scale agreement among many raters. Psycholog.Bull. 76 (5), 378.

Gan, Q., Yu, Y., 2015. Restaurant rating: Industrial standard and word-of-mouth a text mining and multi-dimensional sentiment analysis. Proceedings of the Hawaii International Conference on System Sciences. pp. 1332–1340.

Garcia, D., Zanetti, M., Schweitzer, F., 2013. The role of emotions in contributors activity: A case study on the gentoo community. Proceedings of the International Conference on Cloud and Green Computing. pp. 410–417.

Godbole, N., Srinivasaiah, M., Skiena, S., 2007. Large-scale sentiment analysis for news and blogs. Procceding of the First International AAAI Conference on Weblogs and Social Media.

Graziotin, D., Wang, X., Abrahamsson, P., 2013. Are happy developers more productive? The correlation of affective states of software developers and their self-assessed productivity. Proceedings of the International Conference on Product-Focused Software Process Improvement. pp. 50–64.

Guzman, E., 2013. Visualizing emotions in software development projects. Proceedings of the Conference on Software Visualization. pp. 1–4.

Guzman, E., Azócar, D., Li, Y., 2014. Sentiment analysis of commit comments in github: An empirical study. Proceedings of the International Conference on Mining Software Repositories. pp. 352–355.

Guzman, E., Bruegge, B., 2013. Towards emotional awareness in software development teams. Proceedings of the International Symposium on the Foundations of Software Engineering. pp. 671–674.

Guzman, E., Maalej, W., 2014. How do users like this feature? A fine grained sentiment analysis of app reviews. Proceedings of the International Requirements Engineering Conference. pp. 153–162.

Host, M., Regnell, B., Wohlin, C., 2000. Using students as subjects: A comparative study of students and professionals in lead-time impact assessment. Emp. Softw. Eng. 5 (3).

Hu, M., Liu, B., 2004. Mining and summarizing customer reviews. Proceedings of the International Conference on Knowledge Discovery and Data Mining. pp. 168–177.

Hutto, C., Gilbert, E., 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. Proceedings of the Proceedings of the Eighth International AAAI Conference on Weblogs and Social Media. pp. 216–225.

Islam, M., Zibran, M., 2016. Exploration and exploitation of developers' sentimental variations in software engineering. Intern. J. Softw. Innov. 4 (4), 35–55.

Islam, M., Zibran, M., 2016. Towards understanding and exploiting developers' emotional variations in software engineering. Proceedings of the International Conference on Software Engineering Research Management and Applications. pp. 185–192.

Islam, M., Zibran, M., 2017. A comparison of dictionary building methods for sentiment analysis in software engineering text. Proceedings of the ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. pp. 478–479.

Islam, M., Zibran, M., 2017. Leveraging automated sentiment analysis in software engineering. Proceedings of the Mining Software Repositories. pp. 203–214.

Islam, M., Zibran, M., 2018. A comparison of software engineering domain specific sentiment analysis tools. IEEE International Conference on Software Analysis, Evolution and Reengineering. pp. 487–491.

Islam, M., Zibran, M., 2018. "deva: Sensing emotions in the valence arousal space in software engineering tex. proceedings of the 33rd ACM/SIGAPP Symposium On Applied Computing (SAC). pp. 1536–1543.

Jia, L., Yu, C., Meng, W., 2009. The effect of negation on sentiment analysis and retrieval effectiveness. Proceedings of the ACM Conference on Information and Knowledge Management. pp. 1827–1830.

Jiarpakdee, J., Ihara, A., Matsumoto, K., 2016. Understanding question quality through

affective aspect in Q&A site. Proceedings of the International Workshop on Emotion Awareness in Software Engineering. pp. 12–17.
Jongeling, R., Datta, S., Serebrenik, A., 2015. Choosing your weapons: On sentiment analysis tools for software engineering research. Proceedings of the International Conference on Software Maintenance and Evolution. pp. 531–535.
Jongeling, R., Sarkar, P., Datta, S., Serebrenik, A., 2017. On negative results when using sentiment analysis tools for software engineering research. Emp. Softw. Eng. 1–42.
Kennedy, A., Inkpen, D., 2006. Sentiment classification of movie and product reviews using contextual valence shifters. Comput. Intell. 22 (2), 110–125.
Koto, F., Adriani, M., 2015. A comparative study on twitter sentiment analysis: Which features are good? Proceedings of the International Conference on Applications of Natural Language to Information Systems. pp. 453–457.
L. Gatti, M.G., Turchi, M., 2016. Sentiwords: Deriving a high precision and high coverage lexicon for sentiment analysis. IEEE Trans. Affect. Comput. 7 (4), 409–421.
Lesiuk, T., 2005. The effect of music listening on work performance. Psychology of Music 33 (2), 173–191.
Mäntylä, M., Adams, B., Destefanis, G., Graziotin, D., Ortu, M., 2016. Mining valence, arousal, and dominance – possibilities for detecting burnout and productivity. Proceedings of the International Conference on Mining Software Repositories. pp. 247–258.
Mäntylä, M., Novielli, N., Lanubile, F., Claes, M., Kuutila, M., 2017. Bootstrapping a lexicon for emotional arousal in software engineering. Proceedings of the International Conference on Mining Software Repositories. pp. 1–5.
McDuff, D., Karlson, A., Kapoor, A., Roseway, A., Czerwinski, M., 2012. Affectaura: an intelligent system for emotional memory. Proceedings of the Conference on Human Factors in Computing Systems. pp. 849–858.
Morante, R., Liekens, A., Daelemans, W., 2008. Learning the scope of negation in biomedical texts. Proceedings of the Conference on Empirical Methods in Natural Language Processing. pp. 715–724.
Muhammad, A., Wiratunga, N., Lothian, R., Glassey, R., 2013. Domain-based lexicon enhancement for sentiment analysis. Proceedings of the SGAI International Conference on Artificial Intelligence.
Murgia, A., Ortu, M., Tourani, P., Adams, B., 2017. An exploratory qualitative and quantitative analysis of emotions in issue report comments of open source systems. Emp. Softw. Eng. 1—44.
Murgia, A., Tourani, P., Adams, B., Ortu, M., 2014. Do developers feel emotions? An exploratory analysis of emotions in software artifacts. Proceedings of the International Conference on Mining Software Repositories. pp. 261–271.
Nielsen, F., 2011. A new ANEW: Evaluation of a word list for sentiment analysis in microblogs. Proceedings of the ESWC 2011 Workshop on 'Making Sense of Microposts'. pp. 93–98.
NLTK. Natural Language Toolkit for Sentiment Analysis. http://www.nltk.org/api/nltk.sentiment.html.
Novielli, N.. List of Tools Used in Software Engineering to Detect Emotions. http://www.slideshare.net/nolli82/the-challenges-of-affect-detection-in-the-social-programmer-ecosystem.
Novielli, N., Calefato, F., Lanubile, F., 2014. Towards discovering the role of emotions in stack overflow. Proceedings of the International Workshop on Social Software Engineering. pp. 33–40.
Novielli, N., Calefato, F., Lanubile, F., 2015. The challenges of sentiment detection in the social programmer ecosystem. Proceedings of the International Workshop on Social Software Engineering. pp. 33–40.
Novielli, N., Calefato, F., Lanubile, F., 2018. A gold standard for emotion annotation in stack overflow. Proceedings of the International Conference on Mining Software Repositories. pp. 14–17.
Novielli, N., Girardi, D., Lanubile, F., 2018. A benchmark study on sentiment analysis for software engineering research. Proceedings of the International Conference on Mining Software Repositories. pp. 799–808.
Ortu, M., Adams, B., Destefanis, G., Tourani, P., Marchesi, M., Tonelli, R., 2015. Are bullies more productive? Empirical study of affectiveness vs. issue fixing time. Proceedings of the International Conference on Mining Software Repositories. pp. 303–313.
Ortu, M., Destefanis, G., Counsell, S., Swift, S., Tonelli, R., Marchesi, M., 2016. Arsonists or firefighters? Affectiveness in agile software development. Proceedings of the International Conference on Extreme Programming. pp. 144–155.
Ortu, M., Murgia, A., Destefanis, G., Tourani, P., Tonelli, R., Marchesi, M., Adams, B., 2016. The emotional side of software developers in JIRA. Proceedings of the International Conference on Mining Software Repositories. pp. 480–483.
Panga, B., Lee, L., Vaithyanathan, S., 2002. Thumbs up? Sentiment classification using machine learning techniques. Proceedings of the Conference on Empirical Methods in Natural Language Processing. pp. 79–86.
Panichella, S., Sorbo, A., Guzman, E., Visaggio, C., Canfora, G., Gall, H., 2015. How can i improve my app? Classifying user reviews for software maintenance and evolution. Proceedings of the IEEE International Conference on Software Maintenance and Evolution. pp. 281–290.
Passaro, L., Pollacci, L., Lenci, A., 2015. Item: A vector space model to bootstrap an italian emotive lexicon. Proceedings of the Second Italian Conference on Computational Linguistics CLiC-it. pp. 215–220.
Pletea, D., Vasilescu, B., Serebrenik, A., 2014. Security and emotion: Sentiment analysis of security discussions on github. Proceedings of the International Conference on Mining Software Repositories. pp. 348–351.
Prollochs, N., Feuerriegel, S., Neumann, D., 2016. Detecting negation scopes for financial news sentiment using reinforcement learning. Proceedings of the Hawaii International Conference on System Sciences. pp. 1164–1173.
Qiu, G., Liu, B., Bu, J., Chen, C., 2009. Expanding domain sentiment lexicon through double propagation. Proceedings of the International JointConference on Artifical Intelligence. pp. 1199–1204.
Rahman, M., Roy, C., Keivanloo, I., 2015. Recommending insightful comments for source code using crowd sourced knowledge. Proceedings of the International Working Conference on Source Code Analysis and Manipulation. pp. 81–90.
Reyes, A., Rosso, P., Buscaldi, D., 2012. From humor recognition to irony detection: The figurative language of social media. Data Knowl. Eng. 74, 1–12.
Riloff, E., Qadir, A., Surve, P., Silva, L., Gilbert, N., Huang, R., 2013. Sarcasm as contrast between a positive sentiment and negative situation. Proceedings of the Conference on Empirical Methods in Natural Language Processing. pp. 704–714.
Rousinopoulos, A., Robles, G., Barahona, J., 2014. Sentiment analysis of free/open source developers: preliminary findings from a case study. Revista Eletronica de Sistemas de Informacao 13 (2), 1–21.
SentiStregth-SE. Sentiment Analysis Tool, freely available for download. http://laser.cs.uno.edu/Projects/Projects.html.
SentiStregth-SE. Automatic Domain Independent Tool for Sentiment Analysis. http://sentistrength.wlv.ac.uk.
Sinha, V., 2016. Sentiment Analysis on Java Source Code in Large Sofyware Repositories. Youngstown State University, USA.
Sinha, V., Lazar, A., Sahrif, B., 2016. Analyzing developer sentiment in commit logs. Proceedings of the International Conference on Mining Software Repositories. pp. 520–523.
Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C., Ng, A., Potts, C., 2013. Recursive deep models for semantic compositionality over a sentiment treebank. Conference on Empirical Methods in Natural Language Processing. pp. 1631–1642.
Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C., Ng, A., Potts, C., 2013. Recursive deep models for semantic compositionality over a sentiment treebank. Proceedings of the Conference on Empirical Methods in Natural Language Processing. pp. 1631–1641.
StanfordCoreNLP, last access: June 2018. Stanford Core NLP Sentiment Annotator. http://stanfordnlp.github.io/CoreNLP/sentiment.html.
Thelwall, M., Buckley, K., Paltoglou, G., 2012. Sentiment strength detection for the social web. J. Am. Soc. Info. Sci. Tech. 63(1), 163–173.
Tourani, P., Adams, B., 2016. The impact of human discussions on just-in-time quality assurance. Proceedings of the International Conference on Software Analysis, Evolution, and Reengineering. pp. 189–200.
Tourani, P., Jiang, Y., Adams, B., 2014. Monitoring sentiment in open source mailing lists – exploratory study on the apache ecosystem. Proceedings of the Conference of the Centre for Advanced Studies on Collaborative Research. pp. 34–44.
Warriner, A., Kuperman, V., Brysbaert, M., 2013. Norms of valence, arousal, and dominance for 13,915 english lemmas. Behav.Res.Meth. 45 (4), 1191–1207.
Wilson, T., Wiebe, J., Hoffmann., P., 2009. Recognizing contextual polarity: An exploration of features for phrase-level sentiment analysis. J. Comput. Linguist. 35 (3), 399–433.
Wrobel, M., 2013. Emotions in the software development process. Proceedings of the International Conference on Human System Interaction. pp. 518–523.
Wrobel, M., 2016. Towards the participant observation of emotions in software development teams. Proceedings of the Federated Conference on Computer Science and Information Systems. pp. 1545–1548.
Wu, S., Miller, T., Masanz, J., Coarr, M., Halgrim, S., Carrell, D., Clark, C., 2014. Negation's not solved: Generalizability versus optimizability in clinical natura. PLoS ONE 9 (11), e112774.

Md Rakibul Islam is a third year PhD student in Computer Science department of The University of New Orleans (UNO), Louisiana, USA. He is a member of Laboratory for Software Engineering Research (LaSER) at UNO. His research focuses software engineering, particularly on human aspects and security in software engineering. He like to blend his research areas with information retrieval, data mining and machine learning to extract interesting insights in data. He has co-authored more than 10 papers in different journal and venues that include MSR, SANER, ESEM, SAC and others. Before joining at LaSER, he earned his BSc degree from Khulna University, Bangladesh in 2008 and then worked in different software and telecommunication companies. He has been recognized several times for his excellent service during his tenure period in those companies.

Minhaz F. Zibran is an Assistant Professor at the Department of Computer Science, University of New Orleans, USA. His research interests include various aspects of software engineering with particular focus on the application of source code analysis and manipulation for the detection of code smells, program faults, and vulnerabilities. Minhaz's research also includes human aspects of software engineering encompassing sentiment analysis and its implications on software engineering practices. Minhaz has co-authored many scholarly articles published in ACM and IEEE sponsored international conferences and reputed journals. Minhaz has both teaching and industry experience. He has been a reviewer for reputed journals (e.g., Springer EMSE, Elsevier JSS, IEEE Security & Privacy, Elsevier IST). He has also been actively involved in program committees and organizing committees for international conferences and workshops (e.g., ICDF2C'2018, ICPC'2018, SEMotion'2018, AffectRE'2018, IWSC'2018) in his area of research. He is a professional member of the IEEE Computer Society and ACM SIGSOFT.