Opinion Mining for Software Development: A Systematic Literature Review

BIN LIN, Software Institute - Università della Svizzera italiana, Switzerland
NATHAN CASSEE and ALEXANDER SEREBRENIK, Eindhoven University of Technology, The Netherlands
GABRIELE BAVOTA, Software Institute - Università della Svizzera italiana, Switzerland
NICOLE NOVIELLI, University of Bari, Italy
MICHELE LANZA, Software Institute - Università della Svizzera italiana, Switzerland

Opinion mining, sometimes referred to as sentiment analysis, has gained increasing attention in software engineering (SE) studies. SE researchers have applied opinion mining techniques in various contexts, such as identifying developers' emotions expressed in code comments and extracting users' critics toward mobile apps. Given the large amount of relevant studies available, it can take considerable time for researchers and developers to figure out which approaches they can adopt in their own studies and what perils these approaches entail.

We conducted a systematic literature review involving 185 papers. More specifically, we present (1) well-defined categories of opinion mining-related software development activities, (2) available opinion mining approaches, whether they are evaluated when adopted in other studies, and how their performance is compared, (3) available datasets for performance evaluation and tool customization, and (4) concerns or limitations SE researchers might need to take into account when applying/customizing these opinion mining techniques. The results of our study serve as references to choose suitable opinion mining tools for software development activities and provide critical insights for the further development of opinion mining techniques in the SE domain.

CCS Concepts: • **Software and its engineering** → *Software development process management*; *Software libraries and repositories*; • **Information systems** → **Sentiment analysis;**

Additional Key Words and Phrases: Opinion mining, sentiment analysis, software engineering

ACM Reference format:
Bin Lin, Nathan Cassee, Alexander Serebrenik, Gabriele Bavota, Nicole Novielli, and Michele Lanza. 2022. Opinion Mining for Software Development: A Systematic Literature Review. *ACM Trans. Softw. Eng. Methodol.* 31, 3, Article 38 (March 2022), 41 pages.
https://doi.org/10.1145/3490388

38

We gratefully acknowledge the financial support of the Swiss National Science Foundation for the projects PROBE (SNF Project No. 172799) and SENSOR (SNF-JSPS Project No. 183587).
Authors' addresses: B. Lin, G. Bavota, and M. Lanza, Software Institute - Università della Svizzera italiana, 6900 Lugano, Switzerland; emails: {bin.lin, gabriele.bavota, michele.lanza}@usi.ch; N. Cassee and A. Serebrenik, Eindhoven University of Technology, 5612 AZ Eindhoven, The Netherlands; emails: {n.w.cassee, a.serebrenik}@tue.nl; N. Novielli, University of Bari, 70125 Bari, Italy; email: nicole.novielli@uniba.it.

1049-331X/2022/03-ART38 $15.00
https://doi.org/10.1145/3490388

1 INTRODUCTION

Opinion mining, the term coined by Dave et al. [53] in 2003, was introduced to refer to *"processing a set of search results for a given item, generating a list of product attributes (quality, features, etc.) and aggregating opinions about each of them (poor, mixed, good)."* They proposed a tool to classify product review sentences according to the polarity of the sentiment expressed, i.e., whether these sentences have a positive or negative connotation. Tasks that capture sentiment polarity are also called "sentiment analysis" in some other studies [142, 170]. Indeed, the terms "opinion mining" and "sentiment analysis" are often used interchangeably [142, 192].

Since its emergence, opinion mining has evolved and is no longer limited to classifying texts into different polarities. For example, Conrad and Schilder [49] analyzed subjectivity (i.e., whether the text is subjective or objective) of online posts when mining opinions from blogs in the legal domain. Hu et al. [91] adopted a text summarization approach, which identifies the most informative sentences, to mine opinions from online hotel reviews. These new perspectives call for a broader definition of opinion mining. According to Liu [141], *"opinion mining analyzes people's opinions, appraisals, attitudes, and emotions toward entities, individuals, issues, events, topics, and their attributes."*

In recent years, opinion mining has attracted considerable attention from software engineering researchers. Studies have seen the usage of opinion mining in collecting informative app reviews to understand how developers can improve their products and revise their release plans [44, 97, 152, 193, 213, 238]. Researchers have also applied opinion mining techniques to monitor developers' emotions expressed during development activities [41, 81, 127, 168, 189, 221, 244]. Opinion mining has also been used to assess the quality of software products [29, 54].

Given all these studies, it is important to have an overview of existing opinion mining techniques and their applications in software engineering. In this way, researchers can have a base to advance the field, and tool users can better understand how they can apply the existing techniques and what their limitations are.

We provide a systematic literature review on opinion mining for software development activities. Our contributions are: (1) We provide an overview of opinion mining techniques researchers and developers can use for specific tasks; (2) We present datasets developers can use to train or validate techniques; (3) We report on the results of the tool performance validation, which can serve as a guidance for researchers to conduct performance evaluation and sheds light on the threats when using these tools; (4) We identify the common issues software engineering researchers face when applying opinion mining and indicate the potential solutions; (5) We identify directions for future research in the field.

1.1 Scope of Our Study

Opinion mining is evolving and covers a wide range of topics. Adopting the categories by Pang and Lee [192], we consider under the umbrella of works related to opinion mining in software development activities those dealing with:

- **Sentiment polarity identification**, to classify the opinions expressed in the text into one of the distinguishable sentiment polarities (e.g., positive, neutral, or negative). Examples include identifying whether developers are expressing positive sentiment in daily communications and discovering whether a code review is expressing negative aspects, which can be associated with specific shortcomings of the source code.
- **Subjectivity detection and opinion identification**, to decide whether a given text contains subjective opinions or objective information. An example is distinguishing whether developers/users are discussing a fact about software or presenting their own point of view.

- **Joint topic-sentiment analysis**, which considers topics and opinions simultaneously and search for their interactions. For example, researchers might analyze which aspects are mentioned in user reviews (e.g., performance, usability) and whether these discussions are positive or negative.
- **Viewpoints and perspectives identification**, to detect the general attitudes expressed in the texts (e.g., political orientations) instead of detailed opinions toward a specific issue or narrow subject. An example of perspectives include general preferences of some platforms/technologies over others.
- **Other non-factual information identification**, to detect all other types of non-factual information, including, e.g., emotion detection, humor recognition, text genre classification. Tasks such as identifying what requests users are asking for and extracting knowledge embedded in software documents fall into this category.

1.2 Structure of the Article

Section 2 presents the relevant surveys, literature reviews, and mapping studies. Section 3 presents our research questions and our methodology to conduct the systematic literature review. Section 4 reports the results we obtained. Section 5 discusses the replicability of selected primary studies and the impact of how snowballing is conducted. Section 6 discusses the threats that could affect the validity of our results. Section 7 concludes the article.

2 RELATED WORK

Given the development of opinion mining techniques, many secondary studies have been conducted to present an overview of this field. In the following, we discuss relevant **systematic literature reviews (SLR)**, surveys, and mapping studies on opinion mining.

2.1 Secondary Studies of Opinion Mining in General Domains

As one of the earliest secondary study of opinion mining, Liu [143] defined the problem of opinion mining and presented the key tasks and their corresponding techniques in the literature. This study also specifically discussed the issue of spam detection and quality assessment of online reviews. The survey by Ravi and Ravi [207] presented opinion mining tasks, and relevant techniques at a more fine-grained level. That is, all the tasks and subtasks were discussed in the following aspects: the addressed problem, used dataset, selected features, techniques and their performance.

Hemmatian and Sohrabi [88] mainly focused on the categorization of opinion mining techniques. In their survey, opinion mining was classified into four levels: document, sentence, aspect, and concept. They also summarized different types of techniques used in two major opinion mining tasks: aspect extraction and opinion classification. Li et al. [132] classified opinion mining techniques for social multimedia into three categories based on the source of opinions: textual sentiment analysis (mining opinions from social media messages), visual sentiment analysis (mining opinions from visual content such as images and videos), and multimodal sentiment analysis (mining opinions from both textual and visual content).

Instead of presenting opinion mining tasks and techniques, Kumar and Nandkumar [121] identified several challenges in opinion mining from literature, such as non-expertise opinions, spam opinions, and opinion trust worthiness. They also summarized the advantages and disadvantages of current opinion mining techniques.

Mäntylä et al. [155] analyzed the evolution of opinion mining studies. More specifically, they observed the number of relevant publications, the number of citations, and popular publication venues over the years. They also run topic modeling techniques on the papers to obtain a thematic

overview of the research topics. Moreover, they investigated the research topics of the most cited work in this field.

These studies give a good overview of opinion mining tasks and techniques in general domains. However, our previous studies [113, 138, 178] have demonstrated that the performance of sentiment analysis tools trained on the data from other domains (e.g., SentiStrength trained on social media texts) is not satisfactory when they are used in software engineering–related tasks (e.g., identifying sentiment polarity embedded in API discussions). Therefore, a literature review dedicated to the software engineering domain is highly desired.

2.2 Secondary Studies of Opinion Mining in Software Development Activities

We identified eight secondary studies related to opinion mining in software development activities.

The SLR conducted by Sánchez-Gordón and Colomo-Palacios [210] focused on works dealing with emotions of software developers. The authors investigated 66 papers covering 40 discrete emotions expressed by developers and found that while the unreliability of sentiment analysis tools is well recognized, not many works in the literature have leveraged other measures such as self-reported emotions and biometric sensors.

Obaidi and Klünder [184] inspected 80 studies related to sentient polarity and emotion analysis. Their study mainly looked into the application scenarios (i.e., open-source projects, industry, academic), and the motivations (e.g., find best tool, value measurement). They also counted how many times different data types and techniques are used and listed some frequently mentioned difficulties when analyzing sentiment polarity and emotion in software engineering.

Many SLRs have focused on works related to the analysis of app reviews. Martin et al. [158] conducted a survey on papers related to app store analysis and identified the aspects that have been explored as well as research trends. Noei and Lyons [176] surveyed 21 papers, providing guidelines on how to process, analyze, and use user reviews from app stores. Tavakoli et al. [230] investigated the tools developed for analyzing app reviews and presented the types of information these tools can collect and the challenges these tools are facing. Similarly, the SLR by Genc-Nayebi and Abran [74] involved 24 studies and identified techniques for mining online reviews and challenges in the domain. Moreover, they inspected studies concerning the quality assessment of reviews and spam identification.

The remaining two studies fall into the domain of requirements engineering. Meth et al. [163] analyzed 36 publications regarding automated requirements elicitation and classified them based on tool category, degree of automation, knowledge reuse, evaluation approach, and evaluation concepts. Wang et al. [240] provided a systematic mapping study that focuses on leveraging crowdsourced user feedback in requirements engineering. The feedback can be either explicit (e.g., directly given in crowd-generated comments) or implicit (e.g., mined from application logs or usage-generated data). The primary studies surveyed in these two studies often mine opinions and emotions toward software products from user comments and overlap with those related to app review analysis.

While all these studies cover various topics of opinion mining in software development, they have a focus on very specific areas, such as emotions [184, 210], sentiment polarity [184], app review analysis [74, 158, 230], and requirements engineering [163, 240]. Our study aims at giving a complete picture of the usage of opinion mining techniques in software development, focusing on the research questions presented in Section 3.1. While other secondary studies mainly aim to give an overview of current development of the techniques and their applications, we have a different goal, i.e., to help researchers and developers better adopt/customize opinion mining tools in their own work. Therefore, in this literature review, we have specifically looked into the

datasets available, the performance comparison of the available tools, and the issues specific to tool adoption and customization.

3 RESEARCH METHOD

Following the guidelines by Kitchenham and Charters [120] to perform our systematic literature review, we present our research questions, search strategy, study selection process, as well as the methodology for data extraction and analysis.

3.1 Research Questions

To help software engineering researchers better conduct opinion mining related studies and assist practitioners in adopting suitable opinion mining approaches in their projects, this literature review aims to understand the following high-level **research question (RQ)**:

RQ: *How can opinion mining techniques support software development activities?*

To answer this RQ, it is essential to understand what has been accomplished so far with opinion mining techniques in software development activities. Moreover, knowing the limitations of state-of-the-art approaches is needed to improve the existing techniques or propose new approaches. Therefore, to answer this RQ in a more structured manner, we decompose it into the following RQs:

- **RQ$_1$:** *In which software engineering activities has opinion mining been applied?* We aim to understand the application domains of opinion mining techniques in software engineering, to present an overview on how these techniques are used, thus revealing the potential of opinion mining in software-related tasks.
- **RQ$_2$:** *What publicly available opinion mining tools have been adopted/developed to support these activities?* We present the opinion mining techniques proposed in the literature categorized by their functionalities to obtain an overview about which tools can be used for which specific tasks.
- **RQ$_3$:** *How often do researchers evaluate the reliability of opinion mining tools when they adopt the tools out-of-the-box?* As researchers have already disclosed [113, 138, 178], opinion mining techniques might not achieve satisfactory results when applied in a different context than the one they have been designed for. Thus, it is important to assess the reliability of these tools when used out-of-the-box. We investigate how often opinion mining techniques are evaluated before being applied without any customization in software-related studies.
- **RQ$_4$:** *Which opinion mining techniques have been compared in terms of performance and in what contexts?* Since opinion mining tools perform differently in different contexts, we summarize the studies in the literature aimed at comparing the performance of different opinion mining tools in specific contexts. This will quickly point researchers and practitioners to studies aimed at identifying the most appropriate tools to use in a given context.
- **RQ$_5$:** *Which datasets are available for performance evaluation of opinion mining techniques in software-related contexts and how are they curated?* Given that the context might significantly impact the performance of opinion mining tools, we aim to present the available datasets that can be used to either train supervised techniques or validate the tool performance by serving as oracle. To ensure the reliability of the datasets, we only consider the datasets whose correctness has been manually validated by the authors. We exclude datasets that only contain data scraped from online resources without any further sanity check.
- **RQ$_6$:** *What are the concerns raised or the limitations encountered by researchers when using opinion mining techniques?* Our goal is to summarize the issues encountered during the

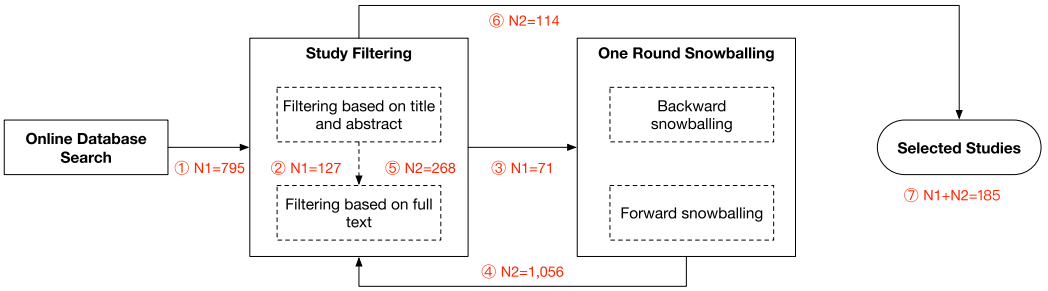

Fig. 1. Relevant study identification process. N1 is the number of papers from the database searching, and N2 is the number of papers resulting from the snowballing.

application of opinion mining techniques in software engineering tasks. We also discuss the potential directions for addressing these issues.

3.2 Relevant Study Identification

The process of identifying relevant studies to be included in our literature review can be seen as Figure 1.

3.2.1 Search Strategy. We used the following digital libraries to search for primary studies: ACM Digital Library [1], IEEE Xplore Digital Library [4], Springer Link Online Library [11], Wiley Online Library [14], Elsevier ScienceDirect [3], and Scopus [9]. We did not include Google Scholar due to several shortcomings identified by Halevi et al. [86], namely, the lack of quality control and clear indexing guidelines, as well as the missing support for data downloads.

The following search query was used to locate primary studies in these online databases:

("**opinion mining**" OR "**sentiment analysis**" OR "**emotion**") AND ("**software**") AND ("**developer**" OR "**development**")

This query has been defined through a trial-and-error procedure performed by the first author and a discussion among all authors. While Landman et al. [126] pointed out that adding an "OR" operator to the query may reduce the number of results in some databases such as IEEE Xplore, we tested such a feature by comparing the results of queries using the "OR" with the aggregated results of several queries each one using one of the search terms in OR. We did not spot any difference, showing that the "OR" operator is working correctly. We conjecture that the difference with the observations of Landman et al. [126] can be attributed to an update of the search engines after their study.

The goal of the query is to retrieve all relevant studies (i.e., high recall) while keeping reasonable the effort needed to remove false positives in the subsequent manual analysis. The search terms "*opinion mining*" and "*sentiment analysis*" have been included, since they are often used as synonyms [140]. Emotion analysis is also attracting attention in studies dealing with human aspects of software engineering [183] and, thus, the term "*emotion*" was included as well. While opinion mining also includes other aspects such as humor detection [26], these topics are not commonly studied in software engineering. Therefore, we do not include the corresponding terms such as "humor" in our query. Concerning the second part of the query, using the term "*software engineering*" to identify relevant studies resulted to be a too strict searching criterion, while only using "*software*" resulted in the introduction of too much noise. The ("*developer*" OR "*development*") search condition allowed to reach a fair balance between the number of papers we need to manually inspect and the coverage of relevant studies. While we are aware that some studies might not

Table 1. Documents Returned by Searching Databases

Source	Returned Documents
ACM Digital Library	340
IEEE Xplore Digital Library	243
Springer Link Online Library	19
Wiley Online Library	29
Elsevier ScienceDirect	46
Scopus	580
Total (excluding duplicates)	795

explicitly include these two terms, this issue has then been mitigated through a snowballing process explained later.

On ACM Digital Library and IEEE Xplore, we conduct the search within its default search box, while in the rest of the databases, due to the large number of irrelevant results returned, we enforced more restrictions when searching. We set the search field of Elsevier ScienceDirect and Scopus to "title, abstract, keywords," meanwhile the "Subject Area" of Scopus was limited to "Computer Science" to exclude studies out of our interest. We only searched "abstract" of Wiley as multiple-field search is not supported. Also in this case "Computer Science" was used to constrain the subject. For SpringerLink Online Library, we set the"Discipline" and "Subdiscipline" to "Computer Science" and "Software Engineering," obtaining 1,967 papers. Since SpringerLink does not allow search on specific fields, we crawled meta-information of these 1,967 papers and filtered them by using our search query in "title, abstract, keywords." We acknowledge that enforcing stricter constraints on some databases might lead to the exclusion of relevant studies. However, the backward and forward snowballing performed later on described significantly mitigates this threat.

3.2.2 Study Selection. Based on the search strategy, we identified relevant studies following a process involving study filtering and snowballing, as indicated in Figure 1. N1 indicates the batch of papers coming from the search query at the end of each step, while N2 shows the number of papers resulting from the snowballing procedure. The lists of papers after each step of study selection can be found in our replication package [136]. Table 1 summarizes the search results. After removing duplicates in the documents returned found in the different databases, we obtained a total of 795 papers.

Study Filtering. The 795 papers went through a two-step filtering process. In the first round, we manually inspected the title and the abstract and removed unrelated documents. A web app was developed to support this process (source code available in our replication package [136]). The web app assigned a batch of papers to filter to each author, who indicated whether it should be (i) "included in the study," (ii) "discarded," or (iii) "used as a secondary study." The last option was used to indicate that the paper was not a primary study, but rather a literature review, survey, or an article introducing the topic. The selected secondary studies have been used in the snowballing process to identify additional primary studies. Each paper was assigned to two of the authors, to reduce the chance that a paper was discarded by mistake. We observed disagreement on 82 papers, which were discussed by all of the authors until a consensus was reached. Then, in the second-round filtering, we downloaded the papers selected as primary studies and each paper was manually inspected by one author to examine if they met our inclusion and exclusion criteria (Table 2).

At the end of first-round filtering, we obtained 127 papers, to include, 662 papers to discard, and 6 papers classified as secondary studies. After the second-round filtering on the 127 papers to include, 71 papers remained as primary studies.

Table 2. Inclusion and Exclusion Criteria

	Inclusion Criteria
IC1	The paper must be peer-reviewed and published at conferences, workshops, or journals; to only include papers that have undergone scrutiny by the scientific community.
IC2	The paper must be accessible online (i.e., PDF files available in the selected databases or through Google Search results); to ensure the accessibility of the studies.
IC3	The paper must be included in one of our databases; to prevent including papers from predatory publishing venues. This criterion only applies to the papers collected from the snowballing process described later.
IC4	The study presented in the paper must be related to software development activities (e.g., requirements, design, implementation, testing, documentation, maintenance, team management); to enforce our research scope listed in Section 1.1.
IC5	The study must adopt at least one opinion mining technique that automatically extracts opinions from texts; to enforce as main research subject "opinion mining."
	Exclusion Criteria
EC1	The paper is not in English. Rationale: English is the primary language for published academic studies.
EC2	The technique presented only works for a language other than English. Rationale: We aim to ensure the techniques in the studies are comparable.
EC3	The paper is a duplicate or a conference paper extended into a journal article. Rationale: We aim to prevent redundancy.
EC4	The paper is not a full research publication (e.g., abstract only submissions, doctoral symposium articles, presentations, tutorials, posters, forewords). Rationale: these artifacts are not subjected to the same peer-reviewing process as full research papers.
EC5	The paper does not describe what approach was used to extract opinions/information. Rationale: Studies lacking such information are often of low quality and do not provide useful information for answering our RQs.

Snowballing. Since keyword-based search might result in omitting relevant studies, we also performed a snowballing-based search on the 127 papers selected as primary studies and on the 6 papers tagged as secondary studies. While 56 out of 127 studies were excluded in the second-round filtering, we still included them in the snowballing process, as they might contain references to papers we are interested in.

We performed both backward and forward snowballing. Backward snowballing was performed during the second-round filtering; each paper was analyzed by one author, and the papers in the references that might be relevant are recorded based on their titles. For forward snowballing, we collected all the papers citing these 133 (127 + 6) papers from Google Scholar. In the end, we obtained 1,056 new papers after duplication removal. All these papers were fed into our paper filtering process. After the first-round filtering, we marked 268 papers as selected primary studies; and after the second-round filtering, 114 papers were left. Due to the limited human resources, we only applied snowballing once instead of iteratively. Therefore, we discarded those papers labeled as "secondary study" identified during our snowballing process, and no further snowballing was performed on them. In total, 185 studies are included in our study.

3.3 Data Extraction and Analysis

To answer the RQ_1–RQ_5 defined in Section 3.1 and facilitate the data extraction process, we used the data extraction form in Table 3 to collect necessary information from the selected studies. This

Table 3. Data Extraction Form

No.	Question	Focus
1	What is the main goal of the whole study?	RQ_1
2	Does the paper propose a new opinion mining approach?	RQ_2
3	Which opinion mining techniques are used (list all of them, clearly stating their name/reference)?	RQ_2
4	Which opinion mining approaches in the paper are publicly available? Write down their name and links. If no approach is publicly available, leave it blank or None.	RQ_2
5	What do the researchers want to achieve by applying the technique(s) (e.g., calculate the sentiment polarity of app reviews)?	RQ_2
6	Is the application context (dataset or application domain) different from that for which the technique was originally designed?	RQ_3
7	Is the performance (precision, recall, run-time, etc.) of the technique verified? If yes, how did they verify it and what are the results?	RQ_3, RQ_4
8	What success metrics are used?	RQ_4
9	Does the paper replicate the results of previous work? If yes, leave a summary of the findings (confirm/partially confirms/contradicts).	RQ_4
10	Which dataset(s) the technique is applied on?	RQ_5
11	Is/Are the dataset(s) publicly available online? If yes, please indicate their name and links.	RQ_5
12	Write down any other comments/notes here.	-

step was conducted together with the "filtering based on full text." A Web app was developed to support this activity (source code available in our replication package [136]), and each paper was manually reviewed by one of the authors.

Given that all extracted information is in free text, we conducted a manual coding process for our data analysis after the data extraction process. This step is important for two reasons: (1) the coding of our extracted information can produce indexes for easing our effort in locating relevant studies, especially considering the large amount of studies we have; (2) the different terminologies used by the authors can be unified, which is essential for answering our RQs.

With the data resulting from the data extraction process, we first identified whether to include the paper by inspecting the answer to No. 12 in Table 3, as we asked the inspectors to take notes here if the paper does not pass the full-text filtering. Then, we identified: (1) the purpose of study (e.g., detecting developers' emotion/sentiment/politeness expressed in software artifacts), (2) whether the approach has been customized, (3) the tools used, (4) whether the approach is available, (5) the type of opinion mining technique (e.g., sentiment polarity analysis), (6) whether the tool is applied in a context different from its origin, (7) & (8) whether the performance of the approach has been verified, (10) the type of dataset (e.g., GitHub issue comments), and (11) whether the dataset is available. As we found that very rarely a study was replicated, we did not collect useful information from No. 9. Not all the information is available for all papers. We used the processes defined in ISO/IEC/IEEE 12207:2017 International Standard [15] for the application domain. More specifically, to identify the relevant process, we compared the purpose of the study to the outcomes, activities, and tasks of each process defined in the ISO/IEC/IEEE 12207:2017 Standard document and then selected the process that matches the best. Additionally, we added the option "team management" to the application domain along with the existing processes in the standard, as it is one of the most popular topics in opinion mining software engineering studies, which focuses on developers instead of specific development processes.

Papers excluded by second-round filtering were also included in the coding process; this is to confirm the decision of exclusion based on ICs and ECs, as each paper was inspected by one author. In total, 395 papers were included in our coding. At first, we selected first 23 papers in our database

for the trial coding (20 out 23 are determined to be included in our study), which was performed by the first two authors. The rest of the authors participated in the discussion until agreement was reached. Then, we equally distributed other 156 papers to all of the authors, namely, on average each author was assigned to 26 different papers for reviewing. As there are several open-ended questions to answer during information retrieval (e.g., what do the researchers want to achieve by applying the technique(s)?), to reduce the duplicate codes written in different ways, we discussed the codes emerged from the output of this round and unified the phrases expressing same meanings. Finally, we equally distributed the remaining 216 papers to all the authors and finished our coding process. The first author double-checked all the coded information and performed the final data organization. While our coded data already provides extensive useful information, we checked the original papers for more detailed information if needed when answering RQ_1–RQ_5.

To answer RQ_6, we inspect the papers proposing or evaluating opinion mining techniques, as we are more interested in the concerns/limitations supported by evidence instead of those based on assumptions. Each paper was manually inspected independently by two of the authors, who extracted insights when the following criteria were satisfied:

- They should be explicitly indicated in the results, discussions, or conclusions.
- They should be relevant to customizing/adopting opinion mining approaches in software engineering.
- They should be supported by data (namely, those proposed without evidence should be discarded).
- They should not describe tool-specific optimizations such as parameter tuning.

We merged the concerns/limitations extracted by the authors and discarded duplicated ones. That is, we removed the same insights from the same paper, but those similar/identical insights were kept if they were extracted from different papers. The extracted insights were then manually categorized based on topic similarity.

4 RESULTS

4.1 RQ_1: In Which Software Engineering Activities Has Opinion Mining Been Applied?

In Table 4, we categorize and summarize the papers that apply opinion mining in software engineering activities.

4.1.1 Design Definition Process. These activities aim to provide detailed information about the elements that can be used to enable the implementation.

Assessing techniques/services for system implementation. Studies have been conducted to mine opinions from online resources to evaluate the strengths and weaknesses of techniques/services. Uddin & Khomh [236] and Lin et al. [137] mined Stack Overflow discussions to extract opinions regarding the pros and cons of adopting certain APIs based on different aspects (e.g., usability, compatibility). Not limited to only APIs, Huang et al. [92] also leveraged Stack Overflow discussions to compare different techniques (e.g., Ant *vs.* Maven). The aspects they used were automatically generated by topic modeling techniques, thus being less structured. Ikram et al. [98] mined tweets to assist in open-source software adoption by analyzing developers' sentiment regarding various aspects. A similar approach has been applied by Ben-Abdallah et al. [32] to online reviews to help developers select proper cloud service.

4.1.2 Knowledge Management Process. These activities aim to provide opportunities to reuse the existing knowledge about development process, skills, and system elements.

Table 4. Software Engineering Activities in Which Opinion Mining Is Applied

Activity	Relevant Papers
Design Definition Process	
Assessing techniques/services for system implementation	[137], [236], [98], [92], [32]
Knowledge Management Process	
Identifying developers' assumptions/rationale from communication	[131], [24]
Mining usage knowledge regarding techniques	[215], [258], [18], [241], [237], [69], [130]
Quality Assurance Process	
Evaluating software quality from crowd source	[206],[90]
Evaluating general user satisfaction	[250], [157], [89], [63], [25], [249], [61], [247]
Evaluating user satisfaction toward specific product aspects	[216], [166],[75], [55], [262], [239], [78], [150], [107], [128], [255], [123], [77], [148], [171], [135], [146], [205], [80], [174], [85], [27], [204], [145], [256], [259], [93], [117]
Identifying issues/requests from developer discussions/issue reports	[227], [191], [165], [115], [64]
Identifying issues/requests/other information from user feedback	[160], [195], [217], [152], [194], [83], [47], [59], [119], [201], [229], [72], [83], [243], [71], [214], [96], [108], [125], [164], [161], [109], [30], [44], [162], [87], [21], [118] [177], [235], [68], [261]
Stakeholder Needs and Requirements Definition Process	
Identifying and evolving requirements from other products	[147], [144], [111], [51]
Identifying and evolving requirements from user feedback	[23], [112], [42], [84], [200], [149], [56], [172], [110], [251], [38], [22], [226], [129]
Identifying requirements from requirement artifacts	[16], [124], [17]
Acquiring deeper understanding of requirements	[203], [219]
Team Management	
Relating emotion/sentiment/politeness to performance/behavior.	[46], [228], [154], [220], [209], [254], [37], [185], [211], [73], [134], [133], [58], [233], [94]
Detecting emotion/sentiment/politeness expressed in software artifacts	[82], [48], [79], [60], [202],[222], [70], [245], [187], [20], [65], [188], [246], [28], [114], [197], [76], [198], [102], [127], [167], [186], [223], [62], [173], [101], [159], [234], [57], [67], [248], [66]
Evaluating the trust among team members	[212], [50]

Identifying developers' assumptions/rationale from communication. Li et al. [131] analyzed discussions from mailing lists to identify assumptions (e.g., a developer guessing what requirements users might have).

This knowledge can be used to infer the rationales behind certain design choices. With similar goals, Alkadhi et al. [24] identified issues, potential solutions, and relevant arguments from development chat messages.

Mining usage knowledge regarding techniques. Several studies have focused on retrieving knowledge about the usage of APIs from online discussions. For example, by analyzing Stack Overflow posts, Uddin et al. [237] documented how APIs are used, and Wang et al. [241] extracted tips for using APIs. Being wary of potential bad programming practice in the automatically retrieved code examples, Serva et al. [215] identified those examples associated with discussions having negative sentiment. Other studies have investigated the negative aspects of APIs. For instance, Zhang and Hou [258] identified discussions on API features from forums that contain negative sentiment. Meanwhile, Ahasanuzzaman et al. [18] and Li et al. [130] identified sentences on Stack Overflow mentioning API issues and negative caveats, respectively. From a coarse-grained level, Fucci et al. [69] classified Stack Overflow posts into 12 types of knowledge, such as functionality, quality, and example.

4.1.3 Quality Assurance Process. These activities aim to identify the issues that might harm software quality and ensure quality requirements are fulfilled. It is worth noting that sometimes

the identified issues during these activities can be further processed to refine the requirements, which is highly relevant to the activities in the category "Stakeholder Needs and Requirements Definition Process." However, in the studies below, such concrete requirement extraction is not conducted.

Evaluating software quality from crowd source. Rahman et al. [206] extracted opinions about quality or issues of the code from Stack Overflow posts to recommend insightful comments for source code. Hu et al. [90] analyzed user comments of the same apps from different platforms to evaluate whether the hybrid development tools, which use a single codebase across platforms, manage to deliver products with similar user-perceived quality.

Evaluating general user satisfaction. Studies have been conducted to understand users' sentiment toward software products by mining their feedback from app reviews [89, 157], tweets [250], or free text reviews from other sources [25, 63]. Durelli et al. [61] took a further step to investigate whether automated tests in mobile apps impact the overall user satisfaction. Some researchers have investigated the sentiment in support tickets [35, 247, 249] to reduce ticket escalations and ensure customer satisfaction. These studies do not look into customer feedback from a more fine-grained perspective (e.g., quality aspects, features).

Evaluating user satisfaction toward specific product aspects. Many studies [75, 78, 80, 85, 93, 107, 123, 145, 148, 150, 171, 174, 204, 216, 239, 255, 256, 259] have classified mobile app reviews into different categories based on the features (e.g., tracking calories), topics (e.g., app theme), or quality aspects (e.g., usability), and then analyzed the sentiment users expressed in these reviews to understand whether the customers are satisfied with the products. A similar technique was also applied to tweets [27, 55, 77, 135, 146, 171], Google research results [262], SourceForge user reviews [205], and online technical review articles [128]. Keertipati et al. [117] converted sentiment toward product features into priorities of mobile app feature development, instead of directly presenting it.

Identifying issues/requests from developer discussions/issue reports. Developer discussions in emails [227] and issue reports [64, 115, 191] have been analyzed to identify bugs and feature requests. Munaiah et al. [165] inspected code reviews to identify possibly missed vulnerabilities.

Identifying issues/requests/other information from user feedback. Researchers have proposed classifiers to cluster mobile app reviews into different categories (e.g., feature request, problem discovery, information seeking, user experiences) [21, 44, 59, 83, 83, 96, 108, 109, 119, 125, 152, 162, 166, 194, 195, 201, 214, 217]. While in different studies the proposed categories can be slightly different, the classified feedback can be further analyzed to identify potential issues, improvement, and new features. A similar approach was applied to tweets [229], user forums [118, 164], and OSS mailing lists [164]. Some studies have specifically focused on identifying types of issues in app reviews [30, 47, 68, 71, 72, 160, 235, 243, 261], while others categorize those reviews into different categories concerning quality (e.g., privacy, usability) or topics without explicitly pointing out the issues [87, 161, 177].

4.1.4 Stakeholder Needs and Requirements Definition Process. These activities aim to help define or refine requirements.

Identifying and evolving requirements from other products. Liu et al. [147] and Jiang et al. [111] mined app descriptions to extract requirements related information and recommend new features, while Liu et al. [144] supported a similar task but only focused on permission-related requirements. Instead of app descriptions, Dalpiaz and Parente [51] analyzed app reviews of competitors to suggest new features.

Identifying and evolving requirements from user feedback. Mobile app reviews are an important source for requirements elicitation. Several studies have mined app reviews to extract

either functional or non-functional requirements [42, 56, 110, 149, 200, 226]. Similar techniques were also applied to reviews in the format of tweets [22, 23, 84, 112, 172, 251], Facebook posts [22], peer-to-peer online review site [38], and feature requests on SourceForge [129].

These activities differ from those to "identify issues/requests/other information from user feedback," as the latter do not aim at eliciting requirements, but rather at assessing the quality of the currently implemented ones.

Identifying requirements from requirement artifacts. Kurtanovic and Maalej [124] trained a classifier to categorize requirements into functional and non-functional (usability, security, operational, performance). Abad et al. [17] proposed an approach to extract text from requirements and identify the non-functional requirements related to usability, operability, and performance. They also implemented a prototype ELICA as a mobile app and conducted a case study to illustrate how it might work in real-life scenarios [16].

Acquiring deeper understanding of requirements. Shi et al. [219] created an approach to classify sentences in feature requests into six different categories (i.e., intent, explanation, benefit, drawback, example, and trivia). Portugal and do Prado Leite [203] used sentiment analysis to extract interdependencies among non-functional requirements, focusing on the relationship between the usability-related requirements as well as the requirements of other quality attributes.

4.1.5 Team Management. These activities aims to understand developers' behavior and performance.

Relating emotion/sentiment/politeness to performance/behavior. The relationship between developers' feelings and their performance or behavior has been widely studied, including the impacts of developers' sentiment, emotions, and attitudes on bug/issue fixing efficiency [58, 94, 185, 254], build success of continuous integration [228], issue reopening [46], routine change [209], activeness of participation [73, 133], likelihood of introducing bugs [233], leadership [37], and productivity [134, 154]. Reversely, the impact of refactoring activities [220] and user feedback [211] on developers' sentiment were also studied.

Detecting emotion/sentiment/politeness expressed in software artifacts. Several researchers have looked into the feelings of developers expressed in various software artifacts. For example, the sentiment polarity detection (i.e., identifying whether a developer is expressing positive or negative feelings) was applied to code review comments [28, 197, 198], emails [20, 65–67, 76, 82, 127, 197, 234], issue reports [57, 60, 82, 114, 186, 197], commit messages [79, 101, 102, 222], commit or pull request comments [202, 223], requirements documents [248], and project reports [159]. Emotions, such as anger, joy, and fear, were detected in issue reports [57, 70, 167, 187, 188] and GitHub comments [173, 245, 246]. Specifically, Elbert et al. [62] detected confusion in code reviews. The politeness of developers was also evaluated in issue reports [57, 186, 187].

Evaluating the trust among team members. Sapkota et al. [212] and Maldonado da Cruz et al. [50] proposed new approaches for estimating trust between developers, leveraging developer interactions and sentiment embedded in pull request or commit comments.

4.2 RQ_2: What Publicly Available Opinion Mining Tools Have Been Adopted/ Developed to Support These Activities?

To answer this RQ, we list all the publicly available opinion mining tools we found in the subject software engineering studies. We classify these tools into two major categories: (1) tools for sentiment polarity/emotion/politeness/trust analysis and (2) tools for artifact content analysis. It is worth noting that while some of these tools are not specifically designed for processing software-related tasks, they are widely used by software engineering researchers. We consider a tool as

Table 5. Opinion Mining Tools for Sentiment Polarity/Emotion/Politeness/Trust Analysis

Technique	Designed for SE data	Based on
Sentiment Polarity Detection		
SentiStrength [232]	No	social media texts
NLTK [95]	No	social media texts
Stanford CoreNLP [225]	No	movie reviews
Watson Natural Language Understanding* [13]	No	unknown
Microsoft Azure Text Analytics* [7]	No	unknown
TextBlob [12]	No	unknown
Affin [175]	No	social media texts
USent [196]	No	TED talk user comments
Syuzhet [5]	No	unknown
Pattern [224]	No	unknown
Rosette* [8]	No	unknown
Aylien* [2]	No	unknown
Narayanan et al., 2013 [169]	No	movie reviews
SentiStrength-SE [106]	Yes	issue reports
Senti4SD [40]	Yes	Stack Overflow posts
SEntiMoji [45]	Yes	issue reports, Stack Overflow posts, code reviews
SentiSW [60]	Yes	issue reports
SentiCR [19]	Yes	code reviews
SentiSE [10]	Yes	code reviews
Emotion Detection		
LIWC* [6]	No	unknown
TensiStrength [231]	No	social media texts
DEVA [105]	Yes	issue reports
MarValous [100]	Yes	Stack Overflow posts, issue reports
EmoTxT [41]	Yes	StackOverflow posts, issue reports
NTUA-SLP [31]	No	social media texts
Politeness Detection		
politeness tool [52]	Yes	Wikipedia and Stack Exchange
Trust Estimation		
Trust-Framework [50]	Yes	GitHub projects

"*" denotes commercial tools.

designed for SE data if it was proposed and evaluated on artifacts generated during software development (e.g., developers' discussions, documentation) by the original authors.

4.2.1 Sentiment Polarity/Emotion/Politeness/Trust Analysis. Tools in this category (Table 5) are mainly used to analyze the feelings expressed by developers. More specifically, sentiment polarity detection tools predict whether a text contains positive, neutral, or negative sentiment. As these tools have been comprehensively compared and evaluated, we kindly invite the readers to refer to Section 4.4 for more information regarding their strengths and weaknesses. Emotion detection tools can extract developers' emotions from the texts, with different tools being able to detect different types of emotions. For example, DEVA [105] and MarValous [100] can detect four emotional states (i.e., excitement, stress, depression, and relaxation), while TensiStrength[1] [231] is used to estimate the strength of stress and relaxation. EmoTxT [41], instead, can detect whether a text contains the following emotions: joy, love, surprise, anger, sadness, and fear. In addition to what EmoTxT [41] is capable to detect, NTUA-SLP [31] can also detect if optimism or pessimism is expressed in the texts, as well as other emotions, i.e., disgust, anticipation, and trust. While these tools in general have good performance, several limitations have been reported. For example, EmoTxT has a relatively low precision and recall for identifying surprise, while NTUA-SLP demonstrated mixed results when predicting the intensity of the emotions. Other issues include the difficulty

[1]TensiStrength can be used with its online demo. Standalone tools is also available for free upon request for academic purposes.

Table 6. Publicly Available Opinion Mining Tools for Artifact Content Analysis

Techniques	Functionality	Based on
LDA [36]	automatically extracts topics from texts	English documents
TwitterLDA [260]	automatically extracts topics from texts	social media texts
ARdoc [195]	identifies app reviews related to information giving, information seeking, feature request, and problem discovery	mobile app reviews
Ticket-Tagger [115]	classifies issue reports into bug, enhancement, and question	issue reports
SURF [226]	identifies app reviews related to information giving, information seeking, feature request, and problem discovery; summarize app reviews based on topics	mobile app reviews
MARC 3.0 [108–110]	identifies app reviews related to bug reports and feature requests; identify topics for functional requests; classify non-functional requests into dependability, reliability, performance, and supportability	mobile app reviews
RE-SWOT [51]	analyzes app reviews to suggest new features	mobile app reviews
DeepTip [241]	extracts API usage tips	Stack Overflow posts
POME [137]	classifies sentences referring to APIs into seven quality aspects (e.g., performance, usability) and determine their sentiment polarity	Stack Overflow posts

of handling negations, irony, and sarcasm (DEVA), processing texts with mixed emotions (TensiStrength), and training on a balanced dataset (MarValous). LIWC [6] calculates the percentage of words falling into 90 different dimensions and summarizes them into four different perspectives: analytical thinking, clout, authenticity, and emotional tone. However, while LIWC is easy to use and provides a broader range of social and psychological insights, the fact of being a commercial software hinders the adaption and further extention of the tool. Currently, there are not many tools available for measuring the politeness of the text (politeness tool [52]). Unlike other tools that take as input texts, Trust-Framework [50] takes a GitHub repository and calculates the trust score among developers. However, the estimated trust scores have not been verified in real team projects.

4.2.2 Artifact Content Analysis. Tools in this category (Table 6) are mainly used to identify the topics or categories of texts from software artifacts. The topics/categories can be either automatically generated (LDA [36] and TwitterLDA [260]) or pre-defined (all the rest). Those tools without pre-defined categories are borrowed from other domains, while the rest are specifically designed for software engineering tasks.

LDA [36] and TwitterLDA [260] are based on Bayesian model. Both of these two tools take a collection of texts as input and output the potential topics of the texts. However, a drawback would be the necessity of knowing the dimension of topics in advance. ARdoc [195], SURF [226], MARC 3.0 [108–110], and RE-SWOT [51] are the tools for user review analysis. More specifically, given the user reviews, these tools can be used to classify the reviews into different categories such as feature request and problem discovery (ARdoc, SURF, MARC 3.0), associate reviews to different topics (SURF), identify different types of non-functional requirements such as performance and usability (MARC 3.0), and extract features classified in strengths, weaknesses, threats, and opportunities (RE-SWOT). While these tools achieve good performance in classifying app reviews based on their categories, several limitations exist. For example, the topic categorization can be coarse-grained (ARdoc). Meanwhile, MARC 3.0 uses only textual information, ignoring other potentially useful meta information such as star ratings and submission time of review. Tools like RE-SWOT do not consider the trend over time, hence the users might not know if some issues have already been addressed. Ticket-Tagger [115] takes GitHub issues and labels them into different categories including bug report, enhancement, and question. However, the recall for enhancement is relatively lower than that of other classes, and there are relatively higher number of false positives for detecting questions. DeepTip [241] and POME [137] both analyzed Stack Overflow posts. The

Table 7. Number of Tools Being Adopted, Used in Different Domains, and How Often Their Performance Is Verified

Tool	# Adopted	# Used Differently (# Verified / # Unverified)
SentiStrength [232]	15	15 (2/13)
politeness tool [52]	5	5 (0/5)
LDA [36]	3	3 (0/3)
NLTK [95]	3	3 (0/3)
LIWC [6]	3	3 (0/3)
Senti4SD [40]	3	3 (1/2)
Stanford CoreNLP [225]	2	2 (0/2)
SentiStrength-SE [106]	2	1 (0/1)
Watson Natural Language Understanding [13]	1	1 (0/1)
Rosette [8]	1	1 (0/1)
TwitterLDA [260]	1	1 (0/1)
SentiSE [10]	1	0 (0/0)
Pattern [224]	1	1 (0/1)
Aylien [2]	1	1 (0/1)
Syuzhet [5]	1	1 (0/1)
EmoTxT [41]	1	0 (0/0)

former extracted tips on API usage, while the latter categorizes API-related sentences into different quality attributes (e.g., performance, compatibility) and sentiment polarities. While both tools achieve high precision, POME reported a relatively low recall for identifying quality attributes.

4.2.3 Extra Information Related to the Opinion Mining Tools. The results presented in this section provide researchers and developers a reference to the tool they might be able to use in their work. However, we acknowledge that readers might want to have a better understanding of these tools. Therefore, we collected the following information from the original papers proposing these tools: (1) the link to the paper; (2) the link to the tool; (3) the input of the tool; (4) the output of the tool; (5) the core technique used in the tool; (6) the advantages of the tool; and (7) the limitations of the tool. This information can be found in the "supplementary results" page of our online replication package [136]. These supplementary results do not include tools for sentiment polarity analysis, as these tools have the same input and output, and they are extensively compared in the literature (as shown in Section 4.4).

4.3 RQ_3: How Often Do Researchers Evaluate the Reliability of Opinion Mining Tools When They Adopt the Tools Out-of-the-box?

As using opinion mining tools from other domains without performance validation might yield unreliable conclusions [113], we are interested to see whether software engineering researchers consider addressing this concern when adopting opinion mining tools developed by others and use them to analyze their data. Table 7 lists the tools adopted by other researchers, how often they are used in a domain different from the one they have been designed for, and how often the performance is verified when it is used in a different domain. Here, a "different domain" refers to the fact that the type of data in the study is different from that used to customize the tool. For example, Stack Overflow posts and mobile app reviews are considered as a different type of data, despite the fact that they both belong to the "software engineering" domain. In this table, we do not count the cases when the tools are only used to compare the performance with other tools and not chosen as the final tool to support software development activities defined in Section 4.1. To obtain the raw data for this RQ (i.e., the list of papers involved in this RQ and their corresponding performance verification information), please visit the "supplementary results" page of our online replication package [136].

Table 8. Performance Comparison of Sentiment Polarity Detection Tools

Compared Tools	Adopted Metric
issue reports	
SentiStrength, SentiStrength-SE, SentiCR, Senti4SD, **SEntiMoji** [45]	overall accuracy
SentiStrength, SentiStrength-SE, SentiCR, Senti4SD [182]	micro-average F1
SentiStrength, Alchemy (Watson NLU), NLTK, Stanford CoreNLP [113]	weighted kappa
SentiStrength, NLTK, Watson NLU, Microsoft Text Analytics API [116]	weighted kappa
SentiStrength, SentiStrength-SE, **SentiSW** [60]	overall accuracy
SentiStrength, **SentiStrength-SE**, NLTK, Stanford CoreNLP [106]	overall F1
SentiStrength, SentiStrength-SE, NLTK, Stanford CoreNLP [139, 153, 208]	overall accuracy
SentiStrength-SE, Senti4SD, EmoTxT [104]	overall accuracy
Stack Overflow posts	
SentiStrength, SentiStrength-SE, SentiCR, Senti4SD, **SEntiMoji** [45]	overall accuracy
SentiStrength, SentiStrength-SE, SentiCR, Senti4SD [182]	micro-average F1
SentiStrength, SentiStrength-SE, SentiCR, Senti4SD [236]	micro-average F1
SentiStrength, SentiStrength-SE, **Senti4SD** [40]	overall F1
SentiStrength, SentiStrength-SE, NLTK, Stanford CoreNLP [139, 153, 208]	overall accuracy
SentiStrength-SE, Senti4SD, EmoTxT [104]	overall accuracy
code reviews	
SentiStrength, SentiStrength-SE, SentiCR, Senti4SD, **SEntiMoji** [45]	overall accuracy
SentiStrength, SentiStrength-SE, SentiCR, Senti4SD [182]	micro-average F1
SentiStrength, SentiStrength-SE, SentiCR, Senti4SD [28]	micro-average F1
SentiStrength, **SentiCR**, NLTK, Afinn, TextBlob, USent, Vivekn [19]	overall accuracy
SentiStrength-SE, Senti4SD, EmoTxT [104]	overall F1
GitHub comments	
SentiStrength, SentiCR, Senti4SD, Alchemy (Watson NLU), NLTK, Stanford CoreNLP [99]	weighted kappa
mobile app reviews	
SentiStrength, SentiStrength-SE, NLTK, Stanford CoreNLP [139, 153, 208]	overall accuracy

Underlined tools has the best performance based on the adopted metric, and tools in bold face are proposed in the literature.

As it can be seen from Table 7, in most of the cases these tools are used in a domain different than the one they have been designed for. What is concerning is that very few researchers try to validate whether these tools can actually produce reliable results in the context of their study. SentiStrength [232] is the most popular opinion mining tool in our subject papers, and of the 15 studies using it in a different context only in 2 cases its performance has been assessed before using it. This is even more problematic, since general-purpose sentiment analysis tools such as SentiStrength have been shown to be unreliable in the software engineering context [113].

4.4 RQ_4: Which Opinion Mining Techniques Have Been Compared in Terms of Performance and in What Contexts?

Tables 8 and 9 present the performance comparisons of sentiment polarity analysis tools and emotion detection tools, respectively. It is worth noting that while EmoTxT [41] is a tool for emotion detection, the comparison in Reference [104] was made by mapping emotion states to sentiment polarity (e.g., joy is considered positive). The studies in the table are categorized based on the data

Table 9. Performance Comparison of Emotion Detection Tools

Compared Tools	Adopted Metric
issue reports	
TensiStrength, **DEVA** [105]	average F1
issue reports + Stack Overflow posts (mixed)	
DEVA, **MarValous** [100]	average F1

Underlined tools has the best performance based on the adopted metric, and tools in bold face are proposed in the literature.

type used in the performance evaluation. The tool with the best performance is underlined and the metric used is also indicated. As artifact content analysis tools often deal with different tasks, their performance cannot be directly compared in most of the cases, therefore, no such comparisons can be found for those publicly available tools.

From Table 8, we can see that, overall, the tools customized on software-related data usually perform better than tools created for general domains. While the performance of different sentiment polarity analysis tools is widely compared on issue reports, Stack Overflow posts, and code reviews, more attention should be given to GitHub comments and mobile app reviews. The performance on these two types of data has not been verified for the latest development of sentiment polarity analysis tool (e.g., SEntiMoji).

While an overall performance comparison result is given in the table, in practice, tool users might have specific focus and preference. For example, when the amount of data is huge, precision might be more important than recall to avoid noise. Another scenario is when analyzing users' complaints from app reviews, a tool that can better identify negative sentiment is preferred. Therefore, to allow readers to check specific metrics, we have aggregated the comparison results for different metrics, which can be found on the "supplementary results" page of our replication package [136].

4.5 RQ_5: Which Datasets Are Available for Performance Evaluation of Opinion Mining Techniques in Software-related Contexts? How Are They Curated?

We present the datasets that can be used by researchers to evaluate or customize opinion mining techniques for software engineering tasks. More specifically, we list in which paper the dataset was presented, which type of dataset were used, the number of data points in the dataset, the categories used in the dataset, and the number of data points falling into each category. We separate these datasets based on what purpose they can be used for: (1) datasets for sentiment polarity/emotion/politeness detection (Table 10) and (2) datasets for sentiment artifact content analysis (Table 11). If several datasets are presented in one paper, then their dataset ID would be formatted as "DSN-X," where X denotes the index of the dataset in the paper (e.g., DS9-1 refers to the first dataset in paper #9). It is worth noting that we do not include datasets whose download link is no longer valid or whose access needs to be requested to the authors. The original paper of DS1 presents three groups of data, all with manually annotated emotions. However, in the first two groups of the data, only raw annotations were given (i.e., emotions assigned by different annotators) and the conflicts were not resolved. Therefore, here, we only include the data in group 3 in which the conflicts were addressed by the authors.

Most of the datasets included have been manually labeled by at least two evaluators, however, how conflicts were resolved varies. DS1, DS14, DS18-1, DS18-2, DS20, and DS21 were labeled by three evaluators, a label was assigned only when at least two of them agreed, thus no extra process was needed to resolve the disagreements. Similarly, DS5, DS6, DS13, DS15, and DS17 were also

Table 10. Datasets Available for Sentiment Polarity/Emotion/Politeness Detection

Dataset ID	Presented by	Data Type	Data Scale
DS1	Ortu et al., 2016 [190]	JIRA issue comments	4,000 sentences
Data distribution: love (187) / joy (158) / sadness (321) / surprise (28) / anger (340)			
DS2	Ebert et al., 2017 [62]	Gerrit code reviews	792 comments
Data distribution: confusion (156) / no confusion (636)			
DS3	Williams & Mahmoud, 2017 [250]	tweets	1000 tweets
Data distribution: negative (493) / positive (359) / frustration (209) / dissatisfaction (133) / bug report (218) / satisfaction (182) / anticipation (42) / excitement (131)			
DS4	Ahmed et al., 2017 [19]	Gerrit code reviews	1,600 comments
Data distribution: negative (398) / non-negative (1202)			
DS5	Calefato et al., 2018 [40]	Stack Overflow posts	4,423 posts
Data distribution: positive (1527) / negative (1202) / neutral (1694)			
DS6	Novielli et al., 2018 [180]	Stack Overflow posts	4,800 posts
Data distribution: love (1,220) / joy (491) / anger (45) / sadness (882) / fear (230) / surprise (106) / neutral (2,841)			
DS7	Islam & Zibran, 2018 [105]	JIRA issue comments	1,795 comments
Data distribution: excitement (411) / stress (252) / depression (289) / relaxation (227) / neutral (616)			
DS8	Ding et al., 2018 [60]	GitHub comments	3,000 comments
Data distribution: positive (597) / negative (405) / neutral (1,998)			
DS9-1	Lin et al., 2018 [139]	mobile app reviews	341 sentences
Data distribution: positive (186) / negative (130) / neutral (25)			
DS9-2	Lin et al., 2018 [139]	Stack Overflow posts	1,500 sentences
Data distribution: positive (178) / negative (131) / neutral (1,191)			
DS10	Kaur et al., 2018 [116]	JIRA issue comments	500 comments
Data distribution: positive (109) / neutral (226) / negative (53) / contradictory (112)			
DS11	Imtiaz et al., 2018 [99]	Github comments	589 comments
Data distribution: [politeness] polite (194) / neutral (395); [sentiment polarity] positive (93) / neutral (419) / negative (73) / sarcasm (4)			
DS12	Sapkota et al., 2020 [212]	Github comments	616 comments
Data distribution: strongly positive (23) / weakly positive (251) / neutral (194) / weakly negative (126) / strongly negative (22)			

Table 11. Datasets Available for Sentiment Artifact Content Analysis

Dataset ID	Presented by	Data Source	Data Scale
DS13	Chen et al., 2014 [44]	mobile app reviews	12,000 sentences
Data distribution: informative (4212) / non-informative (7788)			
DS14	Maalej et al., 2016 [152]	mobile app reviews	4,400 reviews
Data distribution: bug reports (378) / feature requests (299) / user experiences (737) / ratings (2721)			
DS15	Williams & Mahmoud, 2017 [251]	tweets	4,000 tweets
Data distribution: bug reports (1,061) / user requirements (949) / others (1,990)			
DS16	Liu et al., 2018 [144]	mobile app descriptions	923 sentences
Data distribution (related app permission): contact (208) / record (151) / location (564)			
DS17	Jha & Mahmoud, 2018 [109]	mobile app reviews	2,912 reviews
Data distribution: bug report (1,340) / feature request (801) / other (771)			
DS18-1	Jiang et al., 2019 [111]	mobile app descriptions	533 sentences
Data distribution: dataset labeled with features; statistics not available here due to the large number of features			
DS18-2	Jiang et al., 2019 [111]	mobile app descriptions	2,152 sentences
Data distribution: feature (1,073) / no feature (1,079)			
DS19	Fucci et al., 2019 [69]	API documentation pages	100 pages
Data distribution: functionality (89) / concept (29) / directives (41) / purpose (28) / quality (17) / control (27) / structure (24) / patterns (22) / codeExamples (36) / environment (16) / reference (12) / nonInformation (23)			
DS20	Jha & Mahmoud, 2019 [110]	mobile app reviews	7,100 reviews
Data distribution: dependability (1,252) / usability (1,576) / performance (202) / supportability (677) / miscellaneous (4,024)			
DS21	Wang et al., 2019 [241]	Stack Overflow posts	6566 paragraphs(Para) / 11,379 sentences(Sent)
Data distribution (whether containing API tips): Tip-Para (1,101) / No Tip-Para (5,465) / Tip-Sent (1,110) / No Tip-Sent (10,269)			
DS22	Khan et al., 2019 [118]	Reddit forum posts	712 statements
Data distribution: Feature (46) / Claim-Supporting (211) / Claim-Attacking (129) / Claim-Neutral (175) / Issue (68) / alternative (80)			
DS23	Lin et al., 2019 [137]	Stack Overflow posts	2,165 sentences
Data distribution: community (13) / compatibility (87) / documentation (71) / functional (411) / performance (56) / reliability (87) / usability (230) / none (1,138)			

labeled by three people, but when conflicts emerged after labeling, a majority voting criterion was applied. It is worth noting for DS5, if opposite labels were provided, then the corresponding data point was discarded. DS2, DS4, DS7, DS8, DS11, and DS16 were labeled by 4, 3, 3, 2, 2, and 2 evaluators, respectively. Discussion sessions were held afterwards to determine the final labels for those data points with conflicted labels. DS2 also discarded those data points on which no

agreement could be reached. For DS9-1, DS9-2, and D23, each data point was labeled by two people. When there was a disagreement, the final label was decided by a third person. Similarly, DS19 were labeled by two Ph.D. students, and the conflicts were resolved by two of the authors other than the two students. Four evaluators labeled each data point of DS10, and the dataset used the label "contradictory" to annotate the conflicts. For DS12, the first 100 data points were labeled by two people; as the agreement was reached, the remaining data points were labeled by only one person. We are not able to identify how conflicts were resolved for D3 and D22, while we know that these datasets ware labeled by two evaluators. For D22, the authors mentioned that a guideline (publicly available online) was given to minimize disagreements.

4.6 RQ_6: What Are the Concerns Raised or the Limitations Encountered by Researchers When Using/Customizing Opinion Mining Techniques?

In this RQ, we discuss the concerns and the limitations of using and customizing opinion mining tools for software engineering tasks. Meanwhile, we discuss the potential directions to address these issues.

4.6.1 Using/Customizing Tools for Sentiment Polarity/Emotion/Politeness/Trust Analysis. We identify the following concerns/limitations and potential solutions:

Tool performance is often unsatisfactory. Researchers have found that when applying sentiment polarity and emotion analysis tools on software-related data, the accuracy of their output is often unsatisfactory when the domain of application is not the one the tools have been designed for [99, 113, 116, 139, 211, 258]. What is more concerning is that these tools do not even agree with each other, meaning the results or conclusions might change by applying different tools on the same data [113]. Similar issues also hold for emotion analysis tools [242] and politeness detection tools [99] developed in other domains. Therefore, we recommend that *when adopting opinion mining tools designed for non-software engineering contexts, researchers carefully evaluate the reliability and suitability of these tools, as suggested by References [139, 182].*

One common reason for sentiment polarity misclassification is the domain-specific vocabulary [35, 77, 179]. For example, the occurrence of the word "issue" from issue trackers might mislead the general-domain sentiment analysis tools, and the predictions tend to be more negative than it should be. A possible solution is to *tune the dictionary to include more domain-specific vocabularies [35] or train on software engineering data [19, 250].* Currently, there is a lexicon for emotional arousal in software engineering [156], which can be considered when customizing emotion detection tools for software-related tasks. SentiStrength-SE also provides a list of domain-specific terms containing no sentiments in software engineering context [106], which has been proven more effective than general-domain dictionaries when identifying sentiment polarity in software-related contexts [103]. Another challenge is the detection of irony and sarcasm [77, 105, 106, 236]. Islam and Zibran [106] pointed out that a potential solution is "*combining the dictionary-based lexical method with machine learning*," as done in other domains. The existence of decreasing comparative terms (e.g., little problems) often poses challenges for natural language processing–based techniques [107].

Tool performance varies on different data. Even within the software engineering domain, different datasets can still result in unstable performance of sentiment polarity analysis tools [45, 139, 236]. Similarly, the agreement of the predictions produced by different tools also vary on different datasets [104]. Moreover, even when the data are extracted from the same domain, classifiers might still achieve different performance for different types of text. For example, when detecting confusion in code comments, the comment types (i.e., inline and general comments) can impact the precision and recall [62]. Therefore, especially in the case of supervised techniques, it is

recommended to *leverage datasets from the same data source on which it will be applied when training an approach* [182]. Another fact worth noting is that when extracting emotions, the results are more reliable on sentences expressing "joy" (compared to on sentences expressing "anger") and on team-level aggregated texts (compared to individual texts) [242].

Retraining a tool with software-related data requires substantial effort. As opinion mining tools often do not perform well in software-engineering contexts, researchers sometimes retrain existing approaches with software-related data. However, manually building a training set for supervised approaches (e.g., those based on deep learning) can be exhausting, and it does not guarantee that the retrained approach will get better performance [139]. However, researchers have found that training the model with a mixture of software-related and unrelated data can be a solution: *Pre-training the approach with data from other domains (social media [45], Google News [34], Wikipedia English [208]) can significantly improve the performance.*

Neutral sentiment is difficult to identify. Researchers have found that often neutral texts are mistakenly classified as positive or negative, while the opposite occurs much more rarely [139]. Shen et al. [218] confirmed that both machine learning approaches and lexicon & rule-based approaches have difficulties in correctly identifying neutral texts. Therefore, *when evaluating the performance of a sentiment polarity analysis tool, the dataset containing only positive and negative sentiments are insufficient, since the real challenge comes when neutral items are part of the dataset* [139]. *Applying some balancing techniques (e.g., oversampling and undersampling) might to a certain extent improve the low performance caused by dominant neutral texts [34, 153].*

Human-created gold set for tool customization/evaluation may be unreliable. When creating datasets for tool customization or evaluation, one issue is that sometimes there are no clear guidelines, thus the gold set might contain some noise (e.g., "bug report" is mistakenly labeled as negative) [99, 182]. Another issue is subjectivity during data labeling. Studies have found that when it comes to GitHub comments, people have low agreement regarding sentiment and politeness [99]. Moreover, it is easier for evaluators to agree on emotions, such as love and sadness, than others [167]. Therefore, *clear guidelines are needed for the labeling process, and it is necessary to distinguish the objective report of facts (e.g., there is a bug) from the affective state expressed in the text [182].*

Sentiment polarity is not enough for capturing the attitude. Negative lexicons can also express positive attitudes (e.g., people apologizing for not being able to provide further help shows empathy towards others) [179]. Therefore, when possible, *capturing affective states instead of sentiment polarity might provide more fine-grained information.* However, researchers have also highlighted the increased difficulty in identifying affective states compared to only identifying sentiment polarity [218].

User ratings are not always in line with the sentiment expressed. In app review analysis, user ratings are not reliable as a proxy for the user sentiment. While the reviews with one or two stars are negative in most of the cases, reviews with high ratings may also contain issues [160]. The sentiment expressed in the reviews can be more accurately captured by sentiment analysis tools than star ratings [150].

4.6.2 Using/Customizing Tools for Artifact Content Analysis. We identify the following concerns/limitations and potential solutions: **Single data source may not be enough for mining user feedback.** Researchers have found that tweets provide more objective opinions related to apps compared to reviews on app stores [171]. Besides, software companies often use social media to collect bug reports and feature requests. Thus, tweets, especially those from company support accounts, can be a useful source for mining opinions about software products [195]. Meanwhile, most of the reviews do not contain valuable or actionable information for researchers to improve

their apps [135]. Therefore, it is suggested to *look into different data sources to gather more comprehensive feedback*.

The artifact content can belong to multiple categories. During data labeling, researchers have found that a small portion (around 1.1%) of user reviews is related to more than one type of requirement [149]. Thus, when researchers need to customize an approach, *multi-class classification might be necessary*. As a workaround, splitting the text into multiple parts (e.g., sentences) has also been adopted [149].

Data for training is often unbalanced. When training a classifier for identifying various types of user requests, classifiers usually perform badly on the minority types [119, 129, 149]. *Using both project-specific keywords (e.g., those mined from project description and unlabeled user requests) and non-project-specific keywords (e.g., those derived from requirements ontologies and taxonomies) as features for training classifiers can improve the performance to a certain extent* [129].

The quality of datasets affects the performance of the automatic approach for classifying user reviews. If the size of the training set is small, then traditional machine learning approaches outperform deep learning [229]. Meanwhile, when only the data with highly confident labeling (i.e., two evaluators agree on the same class) are used, the performance of machine learning approaches also improves in review classification [119]. This indicates *the importance of the balance between quantity and quality of the training set.* Another important factor is the annotation guide; when category definitions are misunderstood or apparently have similar meanings, misclassification is more likely to happen [83].

Same words can be used to identify different topics/attributes. When identifying quality attributes mentioned in app reviews, same keywords might correspond to different attributes (e.g., the "fast" in "fast loading" refers to performance, while the "fast" in "the app is easy and I can do things fast with it" is more related to usability) [107]. A potential solution could be "*analyzing keywords more than one term*" [47, 72, 107]. For example, using both bi-grams and tri-grams as features to train classifiers might help correctly classify "fast loading" as performance and classify "do things fast" as "usability." However, this does not guarantee same phrase will not convey several different meanings [72].

The various choices of vocabulary negatively impact the performance of user review classification. Different users might use different keywords and linguistic patterns to explain the same issue, which can lead to review misclassification [47]. One potential way to address this issue is to *include more instances of reviews in the training set [47].* At the same time, errors of spelling and grammatical structure and non-standard sentences can also affect the performance [112], which can be addressed by adding spell checker during preprocessing [83]. Meanwhile, there are vocabulary mismatches between different populations (e.g., the technical vocabulary used by developers vs. informal lexicon in the reviews [47, 148]).

The information provided by users can become invalid due to software evolution. Researchers have noticed that some reviews become outdated, as they describe already removed features or technologies used by the apps, and a potential way to solve this issue is to correlate reviews with the app change logs [162].

Data provided by the source can be incomplete. App reviews provided by Google Play Store are incomplete, and researchers have found that using incomplete reviews might bias the findings. It is recommended to collect user reviews continuously for a long time period [177].

Sentences discussing the interested subjects can be hard to locate. When mining opinions for APIs, the precision drops when the API mention is more than one sentence away from the related opinions or several APIs are mentioned together [236]. For the former, it is recommended also considering four surrounding neighboring sentences as well [258].

5 DISCUSSION

In this section, we discuss the replicability issue of these studies we spotted during the analysis of 185 papers. Additionally, we point out the potential directions for future work.

5.1 Replicability of Selected Studies

During our study, we spotted a few issues that might hinder the replicability of opinion mining-related software engineering studies. First of all, if we take a look at techniques in Section 4.2, then we can easily find that there are much more tools available for sentiment polarity and emotion detection than artifact content analysis, while the latter is also widely used in software-engineering activities (Section 4.1). Indeed, lots of proposed approaches for artifact content analysis are not open-source, which also leads to the fact that researchers are often unable to compare their approach with relevant ones (Section 4.4). Besides, when we extracted available tools and datasets, we found many links in the papers to be invalid, in particular when those artifacts were hosted on personal homepages. Thus, it is recommended to store the artifacts on reliable third-party services such as Zenodo,[2] Figshare,[3] and GitHub.[4] Moreover, the artifacts provided in the paper often lack proper documentation, which makes it hard to comprehend the resources.

5.2 Impact of One-round Snowballing

As snowballing is a very expensive activity, iterative snowballing is rarely performed. However, only conducting a single snowballing round is a threat to the completeness of the relevant primary studies we identified. Therefore, to have a basic idea how many papers we might miss in our study, we randomly sampled 10% of the selected papers (i.e., $\lceil 114 * 0.1 \rceil = 12$) obtained from the first snowballing round and conducted a second-round backward and forward snowballing. After removing the papers already collected in our previous selection process, we got 387 studies (denoted as Set 1). Also, we randomly took 10% of the secondary studies abandoned during our paper selection after first-round snowballing (i.e., $\lceil 11 * 0.1 \rceil = 2$) and inspected whether the cited papers in these studies can be a potential primary study in our literature review. This leads to 24 new studies (denoted as Set 2). We followed the same process as described in Section 3.2.2 to filter these 411 papers based on title and abstract. As a result, we found that 26 studies (25 from Set 1 and 1 from Set 2) might fit into our study scope. The list of papers before and after filtering can be found in the "supplementary data" page of our replication package [136]. When inspecting these 26 papers, we found that 12 are obtained by snowballing a single paper. This fact indicates that if we miss a study addressing a specific issue when conducting keyword-based searching, even if we can include it in the first-round snowballing, then we might still miss many relevant studies. By inspecting the venues of these 26 papers, we found that 11 were not published in software-specific conferences or journals, which made them less likely to be relevant for software-engineering researchers.

This result suggests that iterative snowballing plays an important role in the completeness of selected primary studies. However, many databases currently do not provide a convenient way for automatically collecting the papers during snowballing. We acknowledge this common issue in literature reviews, and we would recommend that the search engines could provide an easy way for researchers to download the citing and cited papers. Meanwhile, our result is only based on a small set of samples, we are not sure if performing snowballing on the rest of the studies will lead to similar amount of new papers, especially when we had the rather extreme case that one paper alone

[2] https://zenodo.org.
[3] https://figshare.com.
[4] http://github.com.

introduced 12 new relevant studies. We would also recommend that future researchers working on literature reviews could conduct similar sampling to provide more quantitative insights on the impact of multi-round snowballing. While our study might not include all relevant studies, the research questions we investigated are not highly dependent on the completeness of the samples. Instead, we believe that given the large number of papers included and the in-depth analysis of these studies, our literature review can still provide valuable information regarding opinion mining in software development.

6 THREATS TO VALIDITY

Wohlin et al. [252] list the potential threats researchers might face during software-engineering research.

Threats to construct validity concern the relation between theory and observation. We only select papers indexed in our chosen databases. There might be relevant studies in other databases, however, we have included the most popular ones. Besides, including search engines like Google Scholar might introduce a large amount of noise including not peer-reviewed work and low-quality papers. Another threat is that the search string might not cover all the studies that fit in our search scope. This is mitigated by our backward and forward snowballing process. We only conducted one-round snowballing, which might still miss some relevant papers. Nevertheless, snowballing requires huge amount of human effort, and conducting a second round can be impractical. We believe that most relevant studies were included based on the expertise that the authors have in this domain. Moreover, the large number of papers included in this study can already bring rich information to readers and answer the research questions with sufficient details. Another threat is that we did not apply extra quality assessment criteria on the primary studies we selected. While quality assessment criteria is sometimes used in literature review studies, many criteria are rather subjective. As we only selected peer-reviewed papers, many papers with major design flaws should have been filtered out. However, we acknowledge that some peer-reviewed studies might still contain significant flaws. Our in-depth analysis of the primary studies through the lens of various research questions can mitigate this issue.

Threats to internal validity concern external factors we did not consider that could affect the variables and the relations being investigated. The databases we used are constantly indexing more papers, and they function like black boxes, meaning we are not able to tell whether their search algorithm would change at some point. However, as we take all the results returned and conducted snowballing, we believe that most relevant papers are included in our study. Another issue is that papers are dynamically indexed in these databases. We might not be able to replicate the search results even if the same search strategy is employed. For example, some papers might be indexed in the database much later than their real publication date. Therefore, it is possible to find more papers in the future even if the publication date range remains unchanged. These factors threaten the replicability of our study.

Threats to external validity concern the generalizability of our findings. We only focused on opinion mining techniques designed for artifacts written in English. While English is used as a "lingua franca" in global software development [151], we acknowledge that developers might create software artifacts (e.g., user interfaces, user manuals) in a language other than English. In fact, researchers have found that industry projects are more likely to contain comments and identifiers in more than one language compared to open-source software projects [199]. Additionally, developers might communicate in other languages as well. As coping with multi-lingual texts remains one of the key challenges in natural language processing, it would be interesting to investigate the relevant studies in the future. Besides, all the selected studies are directly associated with software development processes or developers. This choice was taken, as our goal was assisting researchers

and developers in adopting/customizing relevant approaches in software development activities. Our paper search was performed until early 2020. We acknowledge that additional opinion mining tools and datasets have been released [33, 43, 122, 257] and more performance comparisons have been conducted [39, 43, 181, 253].

Threats to conclusion validity concern the relations between the conclusions and our analyzed data. In our study, each paper was inspected by one author, and the corresponding coding was verified by the first author without further examination due to the large amount of studies in our work. While this did not guarantee the correctness of our coding, we did take extra caution when writing the paper and re-recheck all studies for which something was unclear.

7 CONCLUSIONS

7.1 Summary

In this study, we conducted a systematic literature review involving 185 papers related to opinion mining for software engineering. We first presented fine-grained categories of software development activities in which opinion mining is applied and described what these activities are. We then summarized publicly available opinion mining tools in the subject papers and explained in which context these tools are created. We later investigated whether the performance of these tools are evaluated when adopted in other studies, and we found that very few researchers evaluate tool performance when these tools are used in a domain different from the one they have been designed for. We also presented the contexts in which these tools are compared, so researchers and developers can refer to corresponding studies to figure out which tool might work the best for their own data. We next presented 23 publicly available software-related datasets that can be used to evaluate and customize new opinion mining approaches in the software-engineering domain. In the end, we highlighted the concerns and limitations researchers and developers face when adopting and customizing opinion mining tools in software engineering and indicated potential solutions.

7.2 Insights for Tool Adoption Practices

Our study is by far the largest literature review regarding opinion mining in software development activities, and the results of RQ6 highlight some good practices for using opinion mining tools in this context:

- Use the tool trained and/or evaluated on the same data type of the task.
- When using tools trained on other domain, careful verification of tool performance is necessary.
- Do not expect 100% accuracy of the opinion mining tools, especially when texts contain irony and sarcasm.
- When modeling users' attitude, consider using emotions instead of sentiment polarities. However, be aware that some emotions such as joy are easier to capture than others.
- When collecting users' feedback, aggregate the information from various sources (e.g., Twitter, mobile app stores).
- When analyzing users' reviews, give more weight to the sentiment expressed in the reviews than user ratings, and also pay attention to the validity of the reviews (whether the information is outdated).

7.3 Directions for Future Work

Given the issues we identified for using existing opinion mining tools for software-engineering tasks, we list potential directions for future work, with an aim of advancing this domain.

Opinion mining for other software development activities. While opinion mining has been applied to many software-related tasks, there are still some areas that opinion mining has not set foot into. An example is the application in the human resource management process. Human resource managers and project leaders can mine discussions in open-source project artifacts to understand developers' desired tasks and their capabilities, and this information can be taken into account for recruitment, promotion, and task assignment. Moreover, opinions embedded in user feedback can be leveraged for some more specific tasks, such as identifying the need of ending certain system elements (corresponding to the disposal process defined in ISO/IEC/IEEE 12207:2017 International Standard [15]), as well as selecting the optimal software architecture (corresponding to the architecture definition process) and data structure (corresponding to the design definition process).

Productivity enhancement based on monitored developer feelings. Many studies have investigated the sentiment polarity, emotions, and politeness expressed by developers in software artifacts Section 4.1.5. However, few of them have converted these insights into actionable items. Future researchers could investigate how these measured emotions of developers can be used to enhance productivity. For example, when constant negative emotions are detected from developers, team managers might need to help boost developers' mood and pay more attention to work-life balance. We would expect controlled experiments to evaluate whether the proposed actions are effective.

Performance improvement of sentiment polarity analysis. Inspirations to improve sentiment polarity analysis tools can be distilled from the results in Section 4.6.1. Researchers can focus on constructing vocabularies for specific domains, such as issue reports and app reviews. Also, researchers can integrate several datasets from other domains for pre-training the classifier. As the performance of sentiment analysis tools varies on different datasets, it would also be helpful to design a self-adaptive tool that can adjust the approach based on the type of data it deals with.

Validation of user feedback. Section 4.6.2 pointed out a challenge researchers face when identifying opinions from user feedback, namely, that many opinions are not valid anymore due to software updates. Therefore, it is necessary to propose an approach to distinguish still-valid opinions from outdated ones. This is not trivial, as many feedback are not associated with specific versions, therefore, researchers need to rely on other information such as the published date of the feedback and update logs of software for the classification.

Fine-grained classification of opinion topics. Researchers have managed to identify whether users are expressing requests (e.g., References [109, 152]) or describing issues and extract tips regarding how to use APIs (e.g., Reference [241]). However, these classifications are coarse-grained. Given the large amount of feedback available, it is necessary to further categorize the user feedback to reduce developers' manual effort. Topic-modeling techniques have been used to address this issue (e.g., Reference [92]), however, topics automatically generated by these approaches are sometimes not very meaningful. Some researchers have already tried to define taxonomies for types of app reviews [195]. However, more well-defined taxonomies are needed for other purposes, such as concrete types of API usage tips. Researchers can then classify these opinions in a more fine-grained and meaningful level.

REFERENCES

[1] [n. d.]. ACM Digital Library. Retrieved from https://dl.acm.org/.
[2] [n. d.]. Aylien. Retrieved from https://aylien.com.
[3] [n. d.]. Elsevier ScienceDirect. Retrieved from https://www.sciencedirect.com/.
[4] [n. d.]. IEEE Xplore Digital Library. Retrieved from https://ieeexplore.ieee.org/.
[5] [n. d.]. Introduction to the Syuzhet Package. Retrieved from https://cran.r-project.org/web/packages/syuzhet/vignettes/syuzhet-vignette.html.

[6] [n. d.]. LIWC2015. Retrieved from https://liwc.wpengine.com.
[7] [n. d.]. Microsoft Azure Text Analytics. Retrieved from https://azure.microsoft.com/en-us/services/cognitive-services/text-analytics/.
[8] [n. d.]. Rosette Sentiment Analyzer. Retrieved from https://www.rosette.com/capability/sentiment-analyzer/.
[9] [n. d.]. Scopus. Retrieved from https://www.scopus.com/.
[10] [n. d.]. SentiSE. Retrieved from https://github.com/amiangshu/SentiSE.
[11] [n. d.]. Springer Link Online Library. Retrieved from https://link.springer.com/.
[12] [n. d.]. TextBlob: Simplified Text Processing. Retrieved from https://textblob.readthedocs.io/.
[13] [n. d.]. Watson Natural Language Understanding. Retrieved from https://www.ibm.com/cloud/watson-natural-language-understanding.
[14] [n. d.]. Wiley Online Library. Retrieved from https://onlinelibrary.wiley.com/.
[15] 2017. ISO/IEC/IEEE International Standard - systems and software engineering – software life cycle processes. *ISO/IEC/IEEE 12207:2017(E) First edition 2017-11* (2017). IEEE, 1–157. DOI: https://doi.org/10.1109/IEEESTD.2017.8100771
[16] Zahra Shakeri Hossein Abad, Vincenzo Gervasi, Didar Zowghi, and Ken Barker. 2018. ELICA: An automated tool for dynamic extraction of requirements relevant information. In *Proceedings of the 5th International Workshop on Artificial Intelligence for Requirements Engineering*, Eduard C. Groen, Rachel Harrison, Pradeep K. Murukannaiah, and Andreas Vogelsang (Eds.). IEEE, 8–14. DOI: https://doi.org/10.1109/AIRE.2018.00007
[17] Zahra Shakeri Hossein Abad, Vincenzo Gervasi, Didar Zowghi, and Behrouz H. Far. 2019. Supporting analysts by dynamic extraction and classification of requirements-related knowledge. In *Proceedings of the 41st International Conference on Software Engineering*, Joanne M. Atlee, Tevfik Bultan, and Jon Whittle (Eds.). IEEE/ACM, 442–453. DOI: https://doi.org/10.1109/ICSE.2019.00057
[18] Md. Ahasanuzzaman, Muhammad Asaduzzaman, Chanchal K. Roy, and Kevin A. Schneider. 2020. CAPS: A supervised technique for classifying stack overflow posts concerning API issues. *Empir. Softw. Eng.* 25, 2 (2020), 1493–1532. DOI: https://doi.org/10.1007/s10664-019-09743-4
[19] Toufique Ahmed, Amiangshu Bosu, Anindya Iqbal, and Shahram Rahimi. 2017. SentiCR: A customized sentiment analysis tool for code review interactions. In *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering*, Grigore Rosu, Massimiliano Di Penta, and Tien N. Nguyen (Eds.). IEEE Computer Society, 106–111. DOI: https://doi.org/10.1109/ASE.2017.8115623
[20] Jumoke Abass Alesinloye, Eoin Groarke, Jaganath Babu, Subathra Srinivasan, Greg Curran, and Denis Dennehy. 2019. Sentiment analysis of open source software community mailing list: A preliminary analysis. In *Proceedings of the 15th International Symposium on Open Collaboration*, Björn Lundell, Jonas Gamalielsson, Lorraine Morgan, and Gregorio Robles (Eds.). ACM, 21:1–21:5. DOI: https://doi.org/10.1145/3306446.3340824
[21] Mohamed Ali, Mona Erfani Joorabchi, and Ali Mesbah. 2017. Same app, different app stores: A comparative study. In *Proceedings of the 4th IEEE/ACM International Conference on Mobile Software Engineering and Systems*. IEEE, 79–90. DOI: https://doi.org/10.1109/MOBILESoft.2017.3
[22] Nazakat Ali and Jang-Eui Hong. 2019. Value-oriented requirements: Eliciting domain requirements from social network services to evolve software product lines. *Appl. Sci.* 9, 19 (2019), 3944.
[23] Nazakat Ali, Sangwon Hwang, and Jang-Eui Hong. 2019. Your opinions let us know: Mining social network sites to evolve software product lines. *KSII Trans. Internet Inf. Syst.* 13, 8 (2019), 4191–4211. DOI: https://doi.org/10.3837/tiis.2019.08.021
[24] Rana Alkadhi, Teodora Lata, Emitza Guzman, and Bernd Bruegge. 2017. Rationale in development chat messages: An exploratory study. In *Proceedings of the 14th International Conference on Mining Software Repositories*, Jesús M. González-Barahona, Abram Hindle, and Lin Tan (Eds.). IEEE Computer Society, 436–446. DOI: https://doi.org/10.1109/MSR.2017.43
[25] Asma Musabah Alkalbani, Ahmed Mohamed Ghamry, Farookh Khadeer Hussain, and Omar Khadeer Hussain. 2016. Sentiment analysis and classification for software as a service reviews. In *Proceedings of the 30th IEEE International Conference on Advanced Information Networking and Applications*, Leonard Barolli, Makoto Takizawa, Tomoya Enokido, Antonio J. Jara, and Yann Bocchi (Eds.). IEEE Computer Society, 53–58. DOI: https://doi.org/10.1109/AINA.2016.148
[26] Sareeta Amrute. 2017. Press one for POTUS, two for the German chancellor: Humor, race, and rematerialization in the Indian tech diaspora. *HAU: Ethnogr. Theor.* 7, 1 (2017), 327–352. DOI: https://doi.org/10.14318/hau7.1.023
[27] Rakhmat Arianto, Ford Lumban Gaol, Edi Abdurachman, Yaya Heryadi, Harco Leslie Hendric Spits Warnars, Benfano Soewito, and Horacio Perez-Sanchez. 2017. Quality measurement of Android messaging application based on user experience in microblog. In *Proceedings of the International Conference on Applied Computer and Communication Technologies (ComCom)*. IEEE, 1–5.

[28] Ikram El Asri, Noureddine Kerzazi, Gias Uddin, Foutse Khomh, and Mohammed Amine Janati Idrissi. 2019. An empirical study of sentiments in code reviews. *Inf. Softw. Technol.* 114 (2019), 37–54. DOI: https://doi.org/10.1016/j.infsof.2019.06.005

[29] Issa Atoum. 2020. A novel framework for measuring software quality-in-use based on semantic similarity and sentiment analysis of software reviews. *J. King Saud Univ. - Comput. Inf. Sci.* 32, 1 (2020), 113–125. DOI: https://doi.org/10.1016/j.jksuci.2018.04.012

[30] Elsa Bakiu and Emitza Guzman. 2017. Which feature is unusable? Detecting usability and user experience issues from user reviews. In *Proceedings of the IEEE 25th International Requirements Engineering Conference Workshops*. IEEE Computer Society, 182–187. DOI: https://doi.org/10.1109/REW.2017.76

[31] Christos Baziotis, Athanasiou Nikolaos, Alexandra Chronopoulou, Athanasia Kolovou, Georgios Paraskevopoulos, Nikolaos Ellinas, Shrikanth S. Narayanan, and Alexandros Potamianos. 2018. NTUA-SLP at SemEval-2018 Task 1: Predicting affective content in tweets with deep attentive RNNs and transfer learning. In *Proceedings of the 12th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, 245–255. DOI: https://doi.org/10.18653/v1/s18-1037

[32] Emna Ben-Abdallah, Khouloud Boukadi, Jaime Lloret, and Mohamed Hammami. 2021. CROSA: Context-aware cloud service ranking approach using online reviews based on sentiment analysis. *Concurr. Computat.: Pract. Exper.* 33, 7 (2021), e5358. DOI: https://doi.org/10.1002/cpe.5358

[33] Eeshita Biswas, Mehmet Efruz Karabulut, Lori Pollock, and K. Vijay-Shanker. 2020. Achieving reliable sentiment analysis in the software engineering domain using BERT. In *Proceedings of the IEEE International Conference on Software Maintenance and Evolution (ICSME)*. 162–173. DOI: https://doi.org/10.1109/ICSME46990.2020.00025

[34] Eeshita Biswas, K. Vijay-Shanker, and Lori L. Pollock. 2019. Exploring word embedding techniques to improve sentiment analysis of software engineering texts. In *Proceedings of the 16th International Conference on Mining Software Repositories*, Margaret-Anne D. Storey, Bram Adams, and Sonia Haiduc (Eds.). IEEE/ACM, 68–78. DOI: https://doi.org/10.1109/MSR.2019.00020

[35] Cássio Castaldi Araujo Blaz and Karin Becker. 2016. Sentiment analysis in tickets for IT support. In *Proceedings of the 13th International Conference on Mining Software Repositories*, Miryung Kim, Romain Robbes, and Christian Bird (Eds.). ACM, 235–246. DOI: https://doi.org/10.1145/2901739.2901781

[36] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *J. Mach. Learn. Res.* 3 (2003), 993–1022.

[37] Ian Brooks and Kathleen M. Swigger. 2012. Using sentiment analysis to measure the effects of leaders in global software development. In *Proceedings of the International Conference on Collaboration Technologies and Systems*, Waleed W. Smari and Geoffrey Charles Fox (Eds.). IEEE, 517–524. DOI: https://doi.org/10.1109/CTS.2012.6261099

[38] Jim Buchan, Muneera Bano, Didar Zowghi, and Phonephasouk Volabouth. 2018. Semi-automated extraction of new requirements from online reviews for software product evolution. In *Proceedings of the 25th Australasian Software Engineering Conference*. IEEE Computer Society, 31–40. DOI: https://doi.org/10.1109/ASWEC.2018.00013

[39] Luis Adrián Cabrera-Diego, Nik Bessis, and Ioannis Korkontzelos. 2020. Classifying emotions in stack overflow and JIRA using a multi-label approach. *Knowl.-based Syst.* 195 (2020), 105633. DOI: https://doi.org/10.1016/j.knosys.2020.105633

[40] Fabio Calefato, Filippo Lanubile, Federico Maiorano, and Nicole Novielli. 2018. Sentiment polarity detection for software development. *Empir. Softw. Eng.* 23, 3 (2018), 1352–1382. DOI: https://doi.org/10.1007/s10664-017-9546-9

[41] Fabio Calefato, Filippo Lanubile, and Nicole Novielli. 2017. EmoTxt: A toolkit for emotion recognition from text. In *Proceedings of the 7th International Conference on Affective Computing and Intelligent Interaction Workshops and Demos ('17)*. IEEE Computer Society, 79–80. DOI: https://doi.org/10.1109/ACIIW.2017.8272591

[42] Laura V. Galvis Carreño and Kristina Winbladh. 2013. Analysis of user comments: An approach for software requirements evolution. In *Proceedings of the 35th International Conference on Software Engineering*, David Notkin, Betty H. C. Cheng, and Klaus Pohl (Eds.). IEEE Computer Society, 582–591. DOI: https://doi.org/10.1109/ICSE.2013.6606604

[43] Preetha Chatterjee, Kostadin Damevski, and Lori Pollock. 2021. Automatic extraction of opinion-based Q&A from online developer chats. In *Proceedings of the IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 1260–1272.

[44] Ning Chen, Jialiu Lin, Steven C. H. Hoi, Xiaokui Xiao, and Boshen Zhang. 2014. AR-miner: Mining informative reviews for developers from mobile app marketplace. In *36th International Conference on Software Engineering*, Pankaj Jalote, Lionel C. Briand, and André van der Hoek (Eds.). ACM, 767–778. DOI: https://doi.org/10.1145/2568225.2568263

[45] Zhenpeng Chen, Yanbin Cao, Xuan Lu, Qiaozhu Mei, and Xuanzhe Liu. 2019. SEntiMoji: An emoji-powered learning approach for sentiment analysis in software engineering. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. Association for Computing Machinery, 841–852. DOI: 10.1145/3338906.3338977

[46] Jonathan Cheruvelil and Bruno C. da Silva. 2019. Developers' sentiment and issue reopening. In *Proceedings of the 4th International Workshop on Emotion Awareness in Software Engineering*. IEEE/ACM, 29–33. DOI: https://doi.org/10.1109/SEmotion.2019.00013

[47] Adelina Ciurumelea, Andreas Schaufelbühl, Sebastiano Panichella, and Harald C. Gall. 2017. Analyzing reviews and code of mobile apps for better release planning. In *Proceedings of the IEEE 24th International Conference on Software Analysis, Evolution and Reengineering*, Martin Pinzger, Gabriele Bavota, and Andrian Marcus (Eds.). IEEE Computer Society, 91–102. DOI: https://doi.org/10.1109/SANER.2017.7884612

[48] Maëlick Claes, Mika Mäntylä, and Umar Farooq. 2018. On the use of emoticons in open source software development. In *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. Association for Computing Machinery, 4 pages. DOI: 10.1145/3239235.3267434

[49] Jack G. Conrad and Frank Schilder. 2007. Opinion mining in legal blogs. In *Proceedings of the 11th International Conference on Artificial Intelligence and Law (ICAIL'07)*. ACM, 231–236. DOI: https://doi.org/10.1145/1276318.1276363

[50] Guilherme A. Maldonado da Cruz, Elisa Hatsue Moriya Huzita, and Valéria Delisandra Feltrim. 2016. Estimating trust in virtual teams—A framework based on sentiment analysis. In *Proceedings of the 18th International Conference on Enterprise Information Systems*, Slimane Hammoudi, Leszek A. Maciaszek, Michele Missikoff, Olivier Camp, and José Cordeiro (Eds.). SciTePress, 464–471. DOI: https://doi.org/10.5220/0005830604640471

[51] Fabiano Dalpiaz and Micaela Parente. 2019. RE-SWOT: From user feedback to requirements via competitor analysis. In *25th International Working Conference on Requirements Engineering: Foundation for Software Quality (Lecture Notes in Computer Science, Vol. 11412)*, Eric Knauss and Michael Goedicke (Eds.). Springer, 55–70. DOI: https://doi.org/10.1007/978-3-030-15538-4_4

[52] Cristian Danescu-Niculescu-Mizil, Moritz Sudhof, Dan Jurafsky, Jure Leskovec, and Christopher Potts. 2013. A computational approach to politeness with application to social factors. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. The Association for Computer Linguistics, 250–259. Retrieved from https://www.aclweb.org/anthology/P13-1025/.

[53] Kushal Dave, Steve Lawrence, and David M. Pennock. 2003. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *Proceedings of the 12th International World Wide Web Conference (WWW'03)*. ACM, 519–528. DOI: https://doi.org/10.1145/775152.775226

[54] Rahim Dehkharghani and Cemal Yilmaz. 2013. Automatically identifying a software product's quality attributes through sentiment analysis of tweets. In *Proceedings of the 1st International Workshop on Natural Language Analysis in Software Engineering (NaturaLiSE'13)*. 25–30. DOI: https://doi.org/10.1109/NATuraLiSE.2013.6611717

[55] R. Dehkharghani and C. Yilmaz. 2013. Automatically identifying a software product's quality attributes through sentiment analysis of tweets. In *Proceedings of the 1st International Workshop on Natural Language Analysis in Software Engineering (NaturaLiSE)*. 25–30. DOI: https://doi.org/10.1109/NATuraLiSE.2013.6611717

[56] Roger Deocadez, Rachel Harrison, and Daniel Rodríguez. 2017. Automatically classifying requirements from app stores: A preliminary study. In *Proceedings of the IEEE 25th International Requirements Engineering Conference Workshops*. IEEE Computer Society, 367–371. DOI: https://doi.org/10.1109/REW.2017.58

[57] Giuseppe Destefanis, Marco Ortu, David Bowes, Michele Marchesi, and Roberto Tonelli. 2018. On measuring affects of GitHub issues' commenters. In *Proceedings of the 3rd International Workshop on Emotion Awareness in Software Engineering*, Andrew Begel, Alexander Serebrenik, and Daniel Graziotin (Eds.). ACM, 14–19. DOI: https://doi.org/10.1145/3194932.3194936

[58] Giuseppe Destefanis, Marco Ortu, Steve Counsell, Stephen Swift, Michele Marchesi, and Roberto Tonelli. 2016. Software development: Do good manners matter? *PeerJ Comput. Sci.* 2 (2016), e73. DOI: https://doi.org/10.7717/peerj-cs.73

[59] V. T. Dhinakaran, R. Pulle, N. Ajmeri, and P. K. Murukannaiah. 2018. App review analysis via active learning: Reducing supervision effort without compromising classification accuracy. In *Proceedings of the IEEE 26th International Requirements Engineering Conference (RE)*. 170–181. DOI: https://doi.org/10.1109/RE.2018.00026

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

[61] Vinicius H. S. Durelli, Rafael Serapilha Durelli, André Takeshi Endo, Elder Cirilo, Washington Luiz, and Leonardo C. da Rocha. 2018. Please please me: Does the presence of test cases influence mobile app users' satisfaction? In *Proceedings of the 32nd Brazilian Symposium on Software Engineering*, Uirá Kulesza (Ed.). ACM, 132–141. DOI: https://doi.org/10.1145/3266237.3266272

[62] Felipe Ebert, Fernando Castor, Nicole Novielli, and Alexander Serebrenik. 2017. Confusion detection in code reviews. In *Proceedings of the IEEE International Conference on Software Maintenance and Evolution*. IEEE Computer Society, 549–553. DOI: https://doi.org/10.1109/ICSME.2017.40

[63] Alaa Mustafa El-Halees. 2014. Software usability evaluation using opinion mining. *J. Softw.* 9, 2 (2014), 343–349. DOI: https://doi.org/10.4304/jsw.9.2.343-349

[64] Hassan Fazayeli, Sharifah Mashita Syed-Mohamad, and Nur Shazwani Md Akhir. 2019. Towards auto-labelling issue reports for pull-based software development using text mining approach. *Proced. Comput. Sci.* 161 (2019), 585–592. DOI: https://doi.org/10.1016/j.procs.2019.11.160

[65] Isabella Ferreira, Kate Stewart, Daniel M. Germán, and Bram Adams. 2019. A longitudinal study on the maintainers' sentiment of a large scale open source ecosystem. In *Proceedings of the 4th International Workshop on Emotion Awareness in Software Engineering*. IEEE/ACM, 17–22. DOI: https://doi.org/10.1109/SEmotion.2019.00011

[66] Jennifer Ferreira, Denis Dennehy, Jaganath Babu, and Kieran Conboy. 2019. Winning of hearts and minds: Integrating sentiment analytics into the analysis of contradictions. In *Proceedings of the 18th IFIP WG 6.11 Conference on e-Business, e-Services, and e-Society—Digital Transformation for a Sustainable Society in the 21st Century (Lecture Notes in Computer Science, Vol. 11701)*, Ilias O. Pappas, Patrick Mikalef, Yogesh K. Dwivedi, Letizia Jaccheri, John Krogstie, and Matti Mäntymäki (Eds.). Springer, 392–403. DOI: https://doi.org/10.1007/978-3-030-29374-1_32

[67] Jennifer Ferreira, Michael Glynn, David Hunt, Jaganath Babu, Denis Dennehy, and Kieran Conboy. 2019. Sentiment analysis of open source communities: An exploratory study. In *Proceedings of the 15th International Symposium on Open Collaboration*, Björn Lundell, Jonas Gamalielsson, Lorraine Morgan, and Gregorio Robles (Eds.). ACM, 20:1–20:5. DOI: https://doi.org/10.1145/3306446.3340816

[68] Bin Fu, Jialiu Lin, Lei Li, Christos Faloutsos, Jason I. Hong, and Norman M. Sadeh. 2013. Why people hate your app: Making sense of user feedback in a mobile app store. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Inderjit S. Dhillon, Yehuda Koren, Rayid Ghani, Ted E. Senator, Paul Bradley, Rajesh Parekh, Jingrui He, Robert L. Grossman, and Ramasamy Uthurusamy (Eds.). ACM, 1276–1284. DOI: https://doi.org/10.1145/2487575.2488202

[69] Davide Fucci, Alireza Mollaalizadehbahnemiri, and Walid Maalej. 2019. On using machine learning to identify knowledge in API reference documentation. In *Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, Marlon Dumas, Dietmar Pfahl, Sven Apel, and Alessandra Russo (Eds.). ACM, 109–119. DOI: https://doi.org/10.1145/3338906.3338943

[70] Daviti Gachechiladze, Filippo Lanubile, Nicole Novielli, and Alexander Serebrenik. 2017. Anger and its direction in collaborative software development. In *Proceedings of the 39th IEEE/ACM International Conference on Software Engineering: New Ideas and Emerging Technologies Results Track*. IEEE Computer Society, 11–14. DOI: https://doi.org/10.1109/ICSE-NIER.2017.18

[71] Cuiyun Gao, Jichuan Zeng, David Lo, Chin-Yew Lin, Michael R. Lyu, and Irwin King. 2018. INFAR: Insight extraction from app reviews. In *Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, Gary T. Leavens, Alessandro Garcia, and Corina S. Pasareanu (Eds.). ACM, 904–907. DOI: https://doi.org/10.1145/3236024.3264595

[72] Cuiyun Gao, Wujie Zheng, Yuetang Deng, David Lo, Jichuan Zeng, Michael R. Lyu, and Irwin King. 2019. Emerging app issue identification from user feedback: Experience on WeChat. In *Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice*, Helen Sharp and Mike Whalen (Eds.). IEEE/ACM, 279–288. DOI: https://doi.org/10.1109/ICSE-SEIP.2019.00040

[73] David García, Marcelo Serrano Zanetti, and Frank Schweitzer. 2013. The role of emotions in contributors activity: A case study on the GENTOO community. In *Proceedings of the International Conference on Cloud and Green Computing*. IEEE Computer Society, 410–417. DOI: https://doi.org/10.1109/CGC.2013.71

[74] Necmiye Genc-Nayebi and Alain Abran. 2017. A systematic literature review: Opinion mining studies from mobile app store user reviews. *J. Syst. Softw.* 125 (2017), 207–219.

[75] Xiaodong Gu and Sunghun Kim. 2015. What parts of your apps are loved by users?. In *Proceedings of the 30th IEEE/ACM International Conference on Automated Software Engineering*, Myra B. Cohen, Lars Grunske, and Michael Whalen (Eds.). IEEE Computer Society, 760–770. DOI: https://doi.org/10.1109/ASE.2015.57

[76] Emitza Guzman. 2013. Visualizing emotions in software development projects. In *Proceedings of the 1st IEEE Working Conference on Software Visualization (VISSOFT)*, Alexandru Telea, Andreas Kerren, and Andrian Marcus (Eds.). IEEE Computer Society, 1–4. DOI: https://doi.org/10.1109/VISSOFT.2013.6650529

[77] Emitza Guzman, Rana Alkadhi, and Norbert Seyff. 2017. An exploratory study of Twitter messages about software applications. *Requir. Eng.* 22, 3 (2017), 387–412. DOI: https://doi.org/10.1007/s00766-017-0274-x

[78] Emitza Guzman, Omar Aly, and Bernd Bruegge. 2015. Retrieving diverse opinions from app reviews. In *Proceedings of the ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. IEEE Computer Society, 21–30. DOI: https://doi.org/10.1109/ESEM.2015.7321214

[79] Emitza Guzman, David Azócar, and Yang Li. 2014. Sentiment analysis of commit comments in GitHub: An empirical study. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, Premkumar T. Devanbu, Sung Kim, and Martin Pinzger (Eds.). ACM, 352–355. DOI: https://doi.org/10.1145/2597073.2597118

[80] Emitza Guzman, Padma Bhuvanagiri, and Bernd Bruegge. 2014. FAVe: Visualizing user feedback for software evolution. In *Proceedings of the 2nd IEEE Working Conference on Software Visualization*, Houari A. Sahraoui, Andy Zaidman, and Bonita Sharif (Eds.). IEEE Computer Society, 167–171. DOI: https://doi.org/10.1109/VISSOFT.2014.33

[81] Emitza Guzman and Bernd Bruegge. 2013. Towards emotional awareness in software development teams. In *Proceedings of the Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE'13)*. ACM, 671–674. DOI: https://doi.org/10.1145/2491411.2494578

[82] Emitza Guzman and Bernd Bruegge. 2013. Towards emotional awareness in software development teams. In *Proceedings of the Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, ESEC/FSE'13*, Bertrand Meyer, Luciano Baresi, and Mira Mezini (Eds.). ACM, 671–674. DOI: https://doi.org/10.1145/2491411.2494578

[83] Emitza Guzman, Muhammad El-Haliby, and Bernd Bruegge. 2015. Ensemble methods for app review classification: An approach for software evolution (N). In *Proceedings of the 30th IEEE/ACM International Conference on Automated Software Engineering*, Myra B. Cohen, Lars Grunske, and Michael Whalen (Eds.). IEEE Computer Society, 771–776. DOI: https://doi.org/10.1109/ASE.2015.88

[84] Emitza Guzman, Mohamed Ibrahim, and Martin Glinz. 2017. A little bird told me: Mining tweets for requirements and software evolution. In *Proceedings of the 25th IEEE International Requirements Engineering Conference*, Ana Moreira, João Araújo, Jane Hayes, and Barbara Paech (Eds.). IEEE Computer Society, 11–20. DOI: https://doi.org/10.1109/RE.2017.88

[85] Emitza Guzman and Walid Maalej. 2014. How do users like this feature? A fine grained sentiment analysis of app reviews. In *Proceedings of the IEEE 22nd International Requirements Engineering Conference*, Tony Gorschek and Robyn R. Lutz (Eds.). IEEE Computer Society, 153–162. DOI: https://doi.org/10.1109/RE.2014.6912257

[86] Gali Halevi, Henk Moed, and Judit Bar-Ilan. 2017. Suitability of Google Scholar as a source of scientific information and as a source of data for scientific evaluation: Review of the literature. *J. Inform.* 11, 3 (2017), 823–834. DOI: https://doi.org/10.1016/j.joi.2017.06.005

[87] Majid Hatamian, Jetzabel M. Serna, and Kai Rannenberg. 2019. Revealing the unrevealed: Mining smartphone users privacy perception on app markets. *Comput. Secur.* 83 (2019), 332–353. DOI: https://doi.org/10.1016/j.cose.2019.02.010

[88] Fatemeh Hemmatian and Mohammad Karim Sohrabi. 2019. A survey on classification techniques for opinion mining and sentiment analysis. *Artif. Intell. Rev.* 52, 3 (2019), 1495–1545. DOI: https://doi.org/10.1007/s10462-017-9599-6

[89] Leonard Hoon, Miguel Angel Rodriguez-García, Rajesh Vasa, Rafael Valencia-García, and Jean-Guy Schneider. 2016. App reviews: Breaking the user and developer language barrier. In *Trends and Applications in Software Engineering*, Jezreel Mejia, Mirna Munoz, Álvaro Rocha, and Jose Calvo-Manzano (Eds.). Springer International Publishing, Cham, 223–233.

[90] Hanyang Hu, Shaowei Wang, Cor-Paul Bezemer, and Ahmed E. Hassan. 2019. Studying the consistency of star ratings and reviews of popular free hybrid Android and iOS apps. *Empir. Softw. Eng.* 24, 1 (2019), 7–32. DOI: https://doi.org/10.1007/s10664-018-9617-6

[91] Ya-Han Hu, Yen-Liang Chen, and Hui-Ling Chou. 2017. Opinion mining from online hotel reviews—A text summarization approach. *Inf. Process. Manag.* 53, 2 (2017), 436–449. DOI: https://doi.org/10.1016/j.ipm.2016.12.002

[92] Yi Huang, Chunyang Chen, Zhenchang Xing, Tian Lin, and Yang Liu. 2018. Tell them apart: Distilling technology differences from crowd-scale comparison discussions. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, Marianne Huchard, Christian Kästner, and Gordon Fraser (Eds.). ACM, 214–224. DOI: https://doi.org/10.1145/3238147.3238208

[93] Johannes Huebner, Remo Manuel Frey, Christian Ammendola, Elgar Fleisch, and Alexander Ilic. 2018. What people like in mobile finance apps: An analysis of user reviews. In *Proceedings of the 17th International Conference on Mobile and Ubiquitous Multimedia*, Slim Abdennadher and Florian Alt (Eds.). ACM, 293–304. DOI: https://doi.org/10.1145/3282894.3282895

[94] Syed Fatiul Huq, Ali Zafar Sadiq, and Kazi Sakib. 2019. Understanding the effect of developer sentiment on fix-inducing changes: An exploratory study on GitHub pull requests. In *Proceedings of the 26th Asia-Pacific Software Engineering Conference*. IEEE, 514–521. DOI: https://doi.org/10.1109/APSEC48747.2019.00075

[95] Clayton J. Hutto and Eric Gilbert. 2014. VADER: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the 8th International Conference on Weblogs and Social Media ('14)*. The AAAI Press. Retrieved from http://www.aaai.org/ocs/index.php/ICWSM/ICWSM14/paper/view/8109.

[96] Claudia Iacob, Shamal Faily, and Rachel Harrison. 2016. MARAM: Tool support for mobile app review management. In *Proceedings of the 8th EAI International Conference on Mobile Computing, Applications and Services*, Fahim Kawsar, Pei Zhang, and Mirco Musolesi (Eds.). ACM/ICST, 42–50. DOI: https://doi.org/10.4108/eai.30-11-2016.2266941

[97] Claudia Iacob and Rachel Harrison. 2013. Retrieving and analyzing mobile apps feature requests from online reviews. In *Proceedings of the 10th Working Conference on Mining Software Repositories (MSR'13)*. IEEE Computer Society, 41–44. DOI: https://doi.org/10.1109/MSR.2013.6624001

[98] Muhammad Touseef Ikram, Naveed Anwer Butt, and Muhammad Tanvir Afzal. 2016. Open source software adoption evaluation through feature level sentiment analysis using Twitter data. *Turk. J. Electric. Eng. Comput. Sci.* 24, 5 (2016), 4481–4496.

[99] Nasif Imtiaz, Justin Middleton, Peter Girouard, and Emerson R. Murphy-Hill. 2018. Sentiment and politeness analysis tools on developer discussions are unreliable, but so are people. In *Proceedings of the 3rd International Workshop on Emotion Awareness in Software Engineering*, Andrew Begel, Alexander Serebrenik, and Daniel Graziotin (Eds.). ACM, 55–61. DOI: https://doi.org/10.1145/3194932.3194938

[100] Md Rakibul Islam, Md Kauser Ahmmed, and Minhaz F. Zibran. 2019. MarValous: Machine learning based detection of emotions in the valence-arousal space in software engineering text. In *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, Chih-Cheng Hung and George A. Papadopoulos (Eds.). ACM, 1786–1793. DOI: https://doi.org/10.1145/3297280.3297455

[101] Md Rakibul Islam and Minhaz F. Zibran. 2016. Exploration and exploitation of developers' sentimental variations in software engineering. *Int. J. Softw. Innov.* 4, 4 (2016), 35–55. DOI: https://doi.org/10.4018/IJSI.2016100103

[102] Md Rakibul Islam and Minhaz F. Zibran. 2016. Towards understanding and exploiting developers' emotional variations in software engineering. In *Proceedings of the 14th IEEE International Conference on Software Engineering Research, Management and Applications*, Yeong-Tae Song (Ed.). IEEE Computer Society, 185–192. DOI: https://doi.org/10.1109/SERA.2016.7516145

[103] Md Rakibul Islam and Minhaz F. Zibran. 2017. A comparison of dictionary building methods for sentiment analysis in software engineering text. In *Proceedings of the ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, Ayse Bener, Burak Turhan, and Stefan Biffl (Eds.). IEEE Computer Society, 478–479. DOI: https://doi.org/10.1109/ESEM.2017.67

[104] Md Rakibul Islam and Minhaz F. Zibran. 2018. A comparison of software engineering domain specific sentiment analysis tools. In *Proceedings of the 25th International Conference on Software Analysis, Evolution and Reengineering*, Rocco Oliveto, Massimiliano Di Penta, and David C. Shepherd (Eds.). IEEE Computer Society, 487–491. DOI: https://doi.org/10.1109/SANER.2018.8330245

[105] Md Rakibul Islam and Minhaz F. Zibran. 2018. DEVA: Sensing emotions in the valence arousal space in software engineering text. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, Hisham M. Haddad, Roger L. Wainwright, and Richard Chbeir (Eds.). ACM, 1536–1543. DOI: https://doi.org/10.1145/3167132.3167296

[106] Md Rakibul Islam and Minhaz F. Zibran. 2018. SentiStrength-SE: Exploiting domain specificity for improved sentiment analysis in software engineering text. *J. Syst. Softw.* 145 (2018), 125–146. DOI: https://doi.org/10.1016/j.jss.2018.08.030

[107] S. Jamroonsilp and N. Prompoon. 2013. Analyzing software reviews for software quality-based ranking. In *Proceedings of the 10th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*. 1–6. DOI: https://doi.org/10.1109/ECTICon.2013.6559593

[108] Nishant Jha and Anas Mahmoud. 2017. MARC: A mobile application review classifier. In *Joint Proceedings of REFSQ-2017 Workshops, Doctoral Symposium, Research Method Track, and Poster Track co-located with the 22nd International Conference on Requirements Engineering: Foundation for Software Quality (REFSQ'17) (CEUR Workshop Proceedings, Vol. 1796)*, Eric Knauss, Angelo Susi, David Ameller, Daniel M. Berry, Fabiano Dalpiaz, Maya Daneva, Marian Daun, Oscar Dieste, Peter Forbrig, Eduard C. Groen, Andrea Herrmann, Jennifer Horkoff, Fitsum Meshesha Kifetew, Marite Kirikova, Alessia Knauss, Patrick Maeder, Fabio Massacci, Cristina Palomares, Jolita Ralyté, Ahmed Seffah, Alberto Siena, and Bastian Tenbergen (Eds.). CEUR-WS.org. Retrieved from http://ceur-ws.org/Vol-1796/poster-paper-1.pdf.

[109] Nishant Jha and Anas Mahmoud. 2018. Using frame semantics for classifying and summarizing application store reviews. *Empir. Softw. Eng.* 23, 6 (2018), 3734–3767. DOI: https://doi.org/10.1007/s10664-018-9605-x

[110] Nishant Jha and Anas Mahmoud. 2019. Mining non-functional requirements from App store reviews. *Empir. Softw. Eng.* 24, 6 (2019), 3659–3695. DOI: https://doi.org/10.1007/s10664-019-09716-7

[111] He Jiang, Jingxuan Zhang, Xiaochen Li, Zhilei Ren, David Lo, Xindong Wu, and Zhongxuan Luo. 2019. Recommending new features from mobile app descriptions. *ACM Trans. Softw. Eng. Methodol.* 28, 4 (2019), 22:1–22:29. DOI: https://doi.org/10.1145/3344158

[112] Wei Jiang, Haibin Ruan, Li Zhang, Philip Lew, and Jing Jiang. 2014. For user-driven software evolution: Requirements elicitation derived from mining online reviews. In *Proceedings of the 18th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining (Lecture Notes in Computer Science, Vol. 8444)*, Vincent S. Tseng, Tu Bao Ho, Zhi-Hua Zhou, Arbee L. P. Chen, and Hung-Yu Kao (Eds.). Springer, 584–595. DOI: https://doi.org/10.1007/978-3-319-06605-9_48

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

[114] Francisco Jurado and Pilar Rodríguez Marín. 2015. Sentiment Analysis in monitoring software development processes: An exploratory case study on GitHub's project issues. *J. Syst. Softw.* 104 (2015), 82–89. DOI: https://doi.org/10.1016/j.jss.2015.02.055

[115] Rafael Kallis, Andrea Di Sorbo, Gerardo Canfora, and Sebastiano Panichella. 2019. Ticket tagger: Machine learning driven issue classification. In *Proceedings of the International Conference on Software Maintenance and Evolution*. IEEE, 406–409. DOI: https://doi.org/10.1109/ICSME.2019.00070

[116] A. Kaur, A. P. Singh, G. S. Dhillon, and D. Bisht. 2018. Emotion mining and sentiment analysis in software engineering domain. In *Proceedings of the 2nd International Conference on Electronics, Communication and Aerospace Technology (ICECA)*. 1170–1173. DOI: https://doi.org/10.1109/ICECA.2018.8474619

[117] Swetha Keertipati, Bastin Tony Roy Savarimuthu, and Sherlock A. Licorish. 2016. Approaches for prioritizing feature improvements extracted from app reviews. In *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering*, Sarah Beecham, Barbara A. Kitchenham, and Stephen G. MacDonell (Eds.). ACM, 33:1–33:6. DOI: https://doi.org/10.1145/2915970.2916003

[118] Javed Ali Khan, Yuchen Xie, Lin Liu, and Lijie Wen. 2019. Analysis of requirements-related arguments in user forums. In *Proceedings of the 27th IEEE International Requirements Engineering Conference*, Daniela E. Damian, Anna Perini, and Seok-Won Lee (Eds.). IEEE, 63–74. DOI: https://doi.org/10.1109/RE.2019.00018

[119] Nadeem Al Kilani, Rami Tailakh, and Abualsoud Hanani. 2019. Automatic classification of apps reviews for requirement engineering: Exploring the customers need from healthcare applications. In *Proceedings of the 6th International Conference on Social Networks Analysis, Management and Security*, Mohammad A. Alsmirat and Yaser Jararweh (Eds.). IEEE, 541–548. DOI: https://doi.org/10.1109/SNAMS.2019.8931820

[120] Barbara Kitchenham and Stuart Charters. 2007. Guidelines for Performing Systematic Literature Reviews in Software Engineering. Technical Report. EBSE 2007-001. Keele University and Durham University Joint Report.

[121] Antharasanahalli Venkataramaiah Mohan Kumar and Ambuga Narayanaiyengar Nandkumar. 2020. A survey on challenges and research opportunities in opinion mining. *SN Comput. Sci.* 1, 3 (2020), 171. DOI: https://doi.org/10.1007/s42979-020-00149-4

[122] Anang Kunaefi and Masayoshi Aritsugi. 2021. Extracting arguments based on user decisions in app reviews. *IEEE Access* 9 (2021), 45078–45094.

[123] Binil Kuriachan and Nargis Pervin. 2018. ALDA: An aggregated LDA for polarity enhanced aspect identification technique in mobile app domain. In *Proceedings of the 13th International Conference on Designing for a Digital and Globalized World (Lecture Notes in Computer Science, Vol. 10844)*, Samir Chatterjee, Kaushik Dutta, and Rangaraja P. Sundarraj (Eds.). Springer, 187–204. DOI: https://doi.org/10.1007/978-3-319-91800-6_13

[124] Zijad Kurtanovic and Walid Maalej. 2017. Automatically classifying functional and non-functional requirements using supervised machine learning. In *Proceedings of the 25th IEEE International Requirements Engineering Conference*, Ana Moreira, João Araújo, Jane Hayes, and Barbara Paech (Eds.). IEEE Computer Society, 490–495. DOI: https://doi.org/10.1109/RE.2017.82

[125] Zijad Kurtanovic and Walid Maalej. 2017. Mining user rationale from software reviews. In *Proceedings of the 25th IEEE International Requirements Engineering Conference*, Ana Moreira, João Araújo, Jane Hayes, and Barbara Paech (Eds.). IEEE Computer Society, 61–70. DOI: https://doi.org/10.1109/RE.2017.86

[126] Davy Landman, Alexander Serebrenik, and Jurgen J. Vinju. 2017. Challenges for static analysis of Java reflection: Literature review and empirical study. In *Proceedings of the 39th International Conference on Software Engineering*, Sebastián Uchitel, Alessandro Orso, and Martin P. Robillard (Eds.). IEEE/ACM, 507–518. DOI: https://doi.org/10.1109/ICSE.2017.53

[127] Marc J. Lanovaz and Bram Adams. 2019. Comparing the communication tone and responses of users and developers in two R mailing lists: Measuring positive and negative emails. *IEEE Softw.* 36, 5 (2019), 46–50. DOI: https://doi.org/10.1109/MS.2019.2922949

[128] W. Leopairote, A. Surarerks, and N. Prompoon. 2013. Evaluating software quality in use using user reviews mining. In *Proceedings of the 10th International Joint Conference on Computer Science and Software Engineering (JCSSE)*. 257–262. DOI: https://doi.org/10.1109/JCSSE.2013.6567355

[129] Chuanyi Li, Liguo Huang, Jidong Ge, Bin Luo, and Vincent Ng. 2018. Automatically classifying user requests in crowdsourcing requirements engineering. *J. Syst. Softw.* 138 (2018), 108–123. DOI: https://doi.org/10.1016/j.jss.2017.12.028

[130] Jing Li, Aixin Sun, and Zhenchang Xing. 2018. To do or not to do: Distill crowdsourced negative caveats to augment API documentation. *J. Assoc. Inf. Sci. Technol.* 69, 12 (2018), 1460–1475. DOI: https://doi.org/10.1002/asi.24067

[131] Ruiyin Li, Peng Liang, Chen Yang, Georgios Digkas, Alexander Chatzigeorgiou, and Zhuang Xiong. 2019. Automatic identification of assumptions from the hibernate developer mailing list. In *Proceedings of the 26th Asia-Pacific Software Engineering Conference*. IEEE, 394–401. DOI: https://doi.org/10.1109/APSEC48747.2019.00060

[132] Zuhe Li, Yangyu Fan, Bin Jiang, Tao Lei, and Weihua Liu. 2019. A survey on sentiment analysis and opinion mining for social multimedia. *Multim. Tools Appl.* 78, 6 (2019), 6939–6967. DOI: https://doi.org/10.1007/s11042-018-6445-z

[133] Sherlock Licorish and Stephen MacDonell. 2014. Relating IS developers' attitudes to engagement. In *Proceedings of the 25th Australasian Conference on Information Systems (ACIS'14)*. 1–10.

[134] Sherlock A. Licorish and Stephen G. MacDonell. 2018. Exploring the links between software development task type, team attitudes and task completion performance: Insights from the Jazz repository. *Inf. Softw. Technol.* 97 (2018), 10–25. DOI: https://doi.org/10.1016/j.infsof.2017.12.005

[135] Sherlock A. Licorish, Bastin Tony Roy Savarimuthu, and Swetha Keertipati. 2017. Attributes that predict which features to fix: Lessons for app store mining. In *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*, Emilia Mendes, Steve Counsell, and Kai Petersen (Eds.). ACM, 108–117. DOI: https://doi.org/10.1145/3084226.3084246

[136] Bin Lin, Nathan Cassee, Alexander Serebrenik, Gabriele Bavota, Nicole Novielli, and Michele Lanza. 2021. Replication Package for "Opinion Mining for Software Development: A Systematic Literature Review." DOI: https://doi.org/10.5281/zenodo.5106305

[137] Bin Lin, Fiorella Zampetti, Gabriele Bavota, Massimiliano Di Penta, and Michele Lanza. 2019. Pattern-based mining of opinions in Q&A websites. In *Proceedings of the 41st International Conference on Software Engineering*, Joanne M. Atlee, Tevfik Bultan, and Jon Whittle (Eds.). IEEE/ACM, 548–559. DOI: https://doi.org/10.1109/ICSE.2019.00066

[138] Bin Lin, Fiorella Zampetti, Gabriele Bavota, Massimiliano Di Penta, Michele Lanza, and Rocco Oliveto. 2018. Sentiment analysis for software engineering: How far can we go? In *Proceedings of the 40th International Conference on Software Engineering*, Michel Chaudron, Ivica Crnkovic, Marsha Chechik, and Mark Harman (Eds.). ACM, 94–104. DOI: https://doi.org/10.1145/3180155.3180195

[139] Bin Lin, Fiorella Zampetti, Gabriele Bavota, Massimiliano Di Penta, Michele Lanza, and Rocco Oliveto. 2018. Sentiment analysis for software engineering: How far can we go? In *Proceedings of the 40th International Conference on Software Engineering*, Michel Chaudron, Ivica Crnkovic, Marsha Chechik, and Mark Harman (Eds.). ACM, 94–104. DOI: https://doi.org/10.1145/3180155.3180195

[140] Bing Liu. 2011. Opinion mining and sentiment analysis. In *Web Data Mining*. Springer, 459–526.

[141] Bing Liu. 2011. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data. Second Edition.* Springer. DOI: https://doi.org/10.1007/978-3-642-19460-3

[142] Bing Liu. 2015. *Sentiment Analysis - Mining Opinions, Sentiments, and Emotions.* Cambridge University Press. Retrieved from http://www.cambridge.org/us/academic/subjects/computer-science/knowledge-management-databases-and-data-mining/sentiment-analysis-mining-opinions-sentiments-and-emotions.

[143] Bing Liu and Lei Zhang. 2012. A survey of opinion mining and sentiment analysis. In *Mining Text Data*, Charu C. Aggarwal and ChengXiang Zhai (Eds.). Springer, 415–463. DOI: https://doi.org/10.1007/978-1-4614-3223-4_13

[144] Xueqing Liu, Yue Leng, Wei Yang, Chengxiang Zhai, and Tao Xie. 2018. Mining android app descriptions for permission requirements recommendation. In *Proceedings of the 26th IEEE International Requirements Engineering Conference*, Guenther Ruhe, Walid Maalej, and Daniel Amyot (Eds.). IEEE Computer Society, 147–158. DOI: https://doi.org/10.1109/RE.2018.00024

[145] Yuandong Liu, Yanwei Li, Yanhui Guo, and Miao Zhang. 2016. Stratify mobile app reviews: E-LDA model based on hot "Entity" discovery. In *Proceedings of the 12th International Conference on Signal-Image Technology & Internet-Based Systems*, Kokou Yétongnon, Albert Dipanda, Richard Chbeir, Giuseppe De Pietro, and Luigi Gallo (Eds.). IEEE Computer Society, 581–588. DOI: https://doi.org/10.1109/SITIS.2016.97

[146] Yuzhou Liu, Lei Liu, Huaxiao Liu, and Shanquan Gao. 2020. Combining goal model with reviews for supporting the evolution of apps. *IET Softw.* 14, 1 (2020), 39–49. DOI: https://doi.org/10.1049/iet-sen.2018.5192

[147] Yuzhou Liu, Lei Liu, Huaxiao Liu, and Suji Li. 2019. Information recommendation based on domain knowledge in app descriptions for improving the quality of requirements. *IEEE Access* 7 (2019), 9501–9514. DOI: https://doi.org/10.1109/ACCESS.2019.2891543

[148] Yuzhou Liu, Lei Liu, Huaxiao Liu, and Xiaoyu Wang. 2018. Analyzing reviews guided by App descriptions for the software development and evolution. *J. Softw. Evol. Process.* 30, 12 (2018). DOI: https://doi.org/10.1002/smr.2112

[149] Mengmeng Lu and Peng Liang. 2017. Automatic classification of non-functional requirements from augmented app user reviews. In *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*, Emilia Mendes, Steve Counsell, and Kai Petersen (Eds.). ACM, 344–353. DOI: https://doi.org/10.1145/3084226.3084241

[150] Washington Luiz, Felipe Viegas, Rafael Odon de Alencar, Fernando Mourão, Thiago Salles, Dárlinton B. F. Carvalho, Marcos André Gonçalves, and Leonardo C. da Rocha. 2018. A feature-oriented sentiment rating for mobile app reviews. In *Proceedings of the World Wide Web Conference on World Wide Web*, Pierre-Antoine Champin, Fabien L. Gandon, Mounia Lalmas, and Panagiotis G. Ipeirotis (Eds.). ACM, 1909–1918. DOI: https://doi.org/10.1145/3178876.3186168

[151] B. Lutz. 2009. Linguistic challenges in global software development: Lessons learned in an international SW development division. In *Proceedings of the 4th IEEE International Conference on Global Software Engineering*. 249–253. DOI: https://doi.org/10.1109/ICGSE.2009.33

[152] Walid Maalej, Zijad Kurtanovic, Hadeer Nabil, and Christoph Stanik. 2016. On the automatic classification of app reviews. *Requir. Eng.* 21, 3 (2016), 311–331. DOI: https://doi.org/10.1007/s00766-016-0251-9

[153] Rungroj Maipradit, Hideaki Hata, and Kenichi Matsumoto. 2019. Sentiment classification using N-Gram inverse document frequency and automated machine learning. *IEEE Softw.* 36, 5 (2019), 65–70. DOI: https://doi.org/10.1109/MS.2019.2919573

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

[155] Mika V. Mäntylä, Daniel Graziotin, and Miikka Kuutila. 2018. The evolution of sentiment analysis—A review of research topics, venues, and top cited papers. *Comput. Sci. Rev.* 27 (2018), 16–32. DOI: https://doi.org/10.1016/j.cosrev.2017.10.002

[156] Mika V. Mäntylä, Nicole Novielli, Filippo Lanubile, Maëlick Claes, and Miikka Kuutila. 2017. Bootstrapping a lexicon for emotional arousal in software engineering. In *IEEE/ACM 14th International Conference on Mining Software Repositories (MSR'17)*. 198–202. DOI: 10.1109/MSR.2017.47

[157] Daniel Martens and Timo Johann. 2017. On the emotion of users in app reviews. In *Proceedings of the 2nd IEEE/ACM International Workshop on Emotion Awareness in Software Engineering*. IEEE Computer Society, 8–14. DOI: https://doi.org/10.1109/SEmotion.2017.6

[158] William Martin, Federica Sarro, Yue Jia, Yuanyuan Zhang, and Mark Harman. 2017. A survey of app store analysis for software engineering. *IEEE Trans. Softw. Eng.* 43, 9 (2017), 817–847.

[159] Benjamin Matthies. 2016. Feature-based sentiment analysis of codified project knowledge: A dictionary approach. In *Proceedings of the Pacific Asia Conference On Information Systems (PACIS)*. Association for Information System.

[160] Stuart McIlroy, Nasir Ali, Hammad Khalid, and Ahmed E. Hassan. 2016. Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews. *Empir. Softw. Eng.* 21, 3 (2016), 1067–1106. DOI: https://doi.org/10.1007/s10664-015-9375-7

[161] Iván Tactuk Mercado, Nuthan Munaiah, and Andrew Meneely. 2016. The impact of cross-platform development approaches for mobile applications from the user's perspective. In *Proceedings of the International Workshop on App Market Analytics*, Meiyappan Nagappan, Federica Sarro, and Emad Shihab (Eds.). ACM, 43–49. DOI: https://doi.org/10.1145/2993259.2993268

[162] Montassar Ben Messaoud, Ilyes Jenhani, Nermine Ben Jemaa, and Mohamed Wiem Mkaouer. 2019. A multi-label active learning approach for mobile app user review classification. In *Proceedings of the 12th International Conference on Knowledge Science, Engineering and Management (Lecture Notes in Computer Science, Vol. 11775)*, Christos Douligeris, Dimitris Karagiannis, and Dimitris Apostolou (Eds.). Springer, 805–816. DOI: https://doi.org/10.1007/978-3-030-29551-6_71

[163] Hendrik Meth, Manuel Brhel, and Alexander Maedche. 2013. The state of the art in automated requirements elicitation. *Inf. Softw. Technol.* 55, 10 (2013), 1695–1709.

[164] Itzel Morales-Ramirez, Fitsum Meshesha Kifetew, and Anna Perini. 2019. Speech-acts based analysis for requirements discovery from online discussions. *Inf. Syst.* 86 (2019), 94–112. DOI: https://doi.org/10.1016/j.is.2018.08.003

[165] Nuthan Munaiah, Benjamin S. Meyers, Cecilia O. Alm, Andrew Meneely, Pradeep K. Murukannaiah, Emily Prud'hommeaux, Josephine Wolff, and Yang Yu. 2017. Natural language insights from code reviews that missed a vulnerability. In *Engineering Secure Software and Systems*, Eric Bodden, Mathias Payer, and Elias Athanasopoulos (Eds.). Springer International Publishing, Cham, 70–86.

[166] Sergio Muñoz, Oscar Araque, Antonio F. Llamas, and Carlos Angel Iglesias. 2018. A cognitive agent for mining bugs reports, feature suggestions and sentiment in a mobile application store. In *Proceedings of the 4th International Conference on Big Data Innovations and Applications, Innovate-Data*. IEEE, 17–24. DOI: https://doi.org/10.1109/Innovate-Data.2018.00010

[167] Alessandro Murgia, Marco Ortu, Parastou Tourani, Bram Adams, and Serge Demeyer. 2018. An exploratory qualitative and quantitative analysis of emotions in issue report comments of open source systems. *Empir. Softw. Eng.* 23, 1 (2018), 521–564. DOI: https://doi.org/10.1007/s10664-017-9526-0

[168] Alessandro Murgia, Parastou Tourani, Bram Adams, and Marco Ortu. 2014. Do developers feel emotions? An exploratory analysis of emotions in software artifacts. In *Proceedings of the 11th Working Conference on Mining Software Repositories (MSR'14)*. ACM, 262–271. DOI: https://doi.org/10.1145/2597073.2597086

[169] Vivek Narayanan, Ishan Arora, and Arjun Bhatia. 2013. Fast and accurate sentiment classification using an enhanced naive Bayes model. In *Proceedings of the 14th International Conference on Intelligent Data Engineering and Automated Learning (Lecture Notes in Computer Science, Vol. 8206)*. Springer, 194–201. DOI: https://doi.org/10.1007/978-3-642-41278-3_24

[170] Tetsuya Nasukawa and Jeonghee Yi. 2003. Sentiment analysis: Capturing favorability using natural language processing. In *Proceedings of the 2nd International Conference on Knowledge Capture (K-CAP'03)*, John H. Gennari, Bruce W. Porter, and Yolanda Gil (Eds.). ACM, 70–77. DOI: https://doi.org/10.1145/945645.945658

[171] Maleknaz Nayebi, Henry Cho, and Guenther Ruhe. 2018. App store mining is not enough for app improvement. *Empir. Softw. Eng.* 23, 5 (2018), 2764–2794. DOI: https://doi.org/10.1007/s10664-018-9601-1

[172] M. Nayebi, M. Marbouti, R. Quapp, F. Maurer, and G. Ruhe. 2017. Crowdsourced exploration of mobile app features: A case study of the Fort McMurray wildfire. In *Proceedings of the IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Society Track (ICSE-SEIS)*. 57–66. DOI: https://doi.org/10.1109/ICSE-SEIS.2017.8

[173] Krishna Neupane, Kabo Cheung, and Yi Wang. 2019. EmoD: An end-to-end approach for investigating emotion dynamics in software development. In *Proceedings of the International Conference on Software Maintenance and Evolution*. IEEE, 252–256. DOI: https://doi.org/10.1109/ICSME.2019.00038

[174] Mariaclaudia Nicolai, Luca Pascarella, Fabio Palomba, and Alberto Bacchelli. 2019. Healthcare Android apps: A tale of the customers' perspective. In *Proceedings of the 3rd ACM SIGSOFT International Workshop on App Market Analytics*, Federica Sarro and Maleknaz Nayebi (Eds.). ACM, 33–39. DOI: https://doi.org/10.1145/3340496.3342758

[175] Finn Årup Nielsen. 2011. A new ANEW: Evaluation of a word list for sentiment analysis in microblogs. In *Proceedings of the ESWC2011 Workshop on "Making Sense of Microposts": Big Things Come in Small Packages*. 93–98.

[176] Ehsan Noei and Kelly Lyons. 2019. A survey of utilizing user-reviews posted on Google Play Store. In *Proceedings of the 29th Annual International Conference on Computer Science and Software Engineering*. 54–63.

[177] E. Noei, F. Zhang, and Y. Zou. 2019. Too many user-reviews, what should app developers look at first? *IEEE Trans. Softw. Eng.* 47, 2 (2019), 1–1. DOI: https://doi.org/10.1109/TSE.2019.2893171

[178] Nicole Novielli, Fabio Calefato, Davide Dongiovanni, Daniela Girardi, and Filippo Lanubile. 2020. Can we use SE-specific sentiment analysis tools in a cross-platform setting? In *Proceedings of the IEEE/ACM 17th International Conference on Mining Software Repositories*. 158–168. DOI: https://doi.org/10.1145/3379597.3387446

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

[180] Nicole Novielli, Fabio Calefato, and Filippo Lanubile. 2018. A gold standard for emotion annotation in stack overflow. In *Proceedings of the 15th International Conference on Mining Software Repositories*, Andy Zaidman, Yasutaka Kamei, and Emily Hill (Eds.). ACM, 14–17. DOI: https://doi.org/10.1145/3196398.3196453

[181] Nicole Novielli, Fabio Calefato, Filippo Lanubile, and Alexander Serebrenik. 2021. Assessment of off-the-shelf SE-specific sentiment analysis tools: An extended replication study. *Empir. Softw. Eng.* 26 (2021), 77:1–29.

[182] Nicole Novielli, Daniela Girardi, and Filippo Lanubile. 2018. A benchmark study on sentiment analysis for software engineering research. In *Proceedings of the 15th International Conference on Mining Software Repositories*, Andy Zaidman, Yasutaka Kamei, and Emily Hill (Eds.). ACM, 364–375. DOI: https://doi.org/10.1145/3196398.3196403

[183] Nicole Novielli and Alexander Serebrenik. 2019. Sentiment and emotion in software engineering. *IEEE Softw.* 36, 5 (2019), 6–23.

[184] Martin Obaidi and Jil Klünder. 2021. Development and application of sentiment analysis tools in software engineering: A systematic literature review. In *Proceedings of the Conference on Evaluation and Assessment in Software Engineering*. Association for Computing Machinery, New York, NY, 80–89. DOI: https://doi.org/10.1145/3463274.3463328

[185] Marco Ortu, Bram Adams, Giuseppe Destefanis, Parastou Tourani, Michele Marchesi, and Roberto Tonelli. 2015. Are bullies more productive? Empirical study of affectiveness vs. issue fixing time. In *Proceedings of the 12th IEEE/ACM Working Conference on Mining Software Repositories*, Massimiliano Di Penta, Martin Pinzger, and Romain Robbes (Eds.). IEEE Computer Society, 303–313. DOI: https://doi.org/10.1109/MSR.2015.35

[186] Marco Ortu, Giuseppe Destefanis, Steve Counsell, Stephen Swift, Roberto Tonelli, and Michele Marchesi. 2016. Arsonists or firefighters? Affectiveness in agile software development. In *Proceedings of the 17th International Conference on Agile Processes, in Software Engineering, and Extreme Programming (Lecture Notes in Business Information Processing, Vol. 251)*, Helen Sharp and Tracy Hall (Eds.). Springer, 144–155. DOI: https://doi.org/10.1007/978-3-319-33515-5_12

[187] Marco Ortu, Tracy Hall, Michele Marchesi, Roberto Tonelli, David Bowes, and Giuseppe Destefanis. 2018. Mining communication patterns in software development: A GitHub analysis. In *Proceedings of the 14th International Conference on Predictive Models and Data Analytics in Software Engineering*, Burak Turhan, Ayse Tosun, and Shane McIntosh (Eds.). ACM, 70–79. DOI: https://doi.org/10.1145/3273934.3273943

[188] Marco Ortu, Michele Marchesi, and Roberto Tonelli. 2019. Empirical analysis of affect of merged issues on GitHub. In *Proceedings of the 4th International Workshop on Emotion Awareness in Software Engineering*. IEEE/ACM, 46–48. DOI: https://doi.org/10.1109/SEmotion.2019.00017

[189] Marco Ortu, Alessandro Murgia, Giuseppe Destefanis, Parastou Tourani, Roberto Tonelli, Michele Marchesi, and Bram Adams. 2016. The emotional side of software developers in JIRA. In *Proceedings of the 13th International Conference on Mining Software Repositories (MSR'16)*. ACM, 480–483. DOI: https://doi.org/10.1145/2901739.2903505

[190] Marco Ortu, Alessandro Murgia, Giuseppe Destefanis, Parastou Tourani, Roberto Tonelli, Michele Marchesi, and Bram Adams. 2016. The emotional side of software developers in JIRA. In *Proceedings of the 13th International Conference on Mining Software Repositories*, Miryung Kim, Romain Robbes, and Christian Bird (Eds.). ACM, 480–483. DOI: https://doi.org/10.1145/2901739.2903505

[191] Nitish Pandey, Debarshi Kumar Sanyal, Abir Hudait, and Amitava Sen. 2017. Automated classification of software issue reports using machine learning techniques: An empirical study. *Innov. Syst. Softw. Eng.* 13, 4 (2017), 279–297. DOI: https://doi.org/10.1007/s11334-017-0294-1

[192] Bo Pang and Lillian Lee. 2007. Opinion mining and sentiment analysis. *Found. Trends Inf Retr.* 2, 1–2 (2007), 1–135. DOI: https://doi.org/10.1561/1500000011

[193] Sebastiano Panichella, Andrea Di Sorbo, Emitza Guzman, Corrado Aaron Visaggio, Gerardo Canfora, and Harald C. Gall. 2015. How can I improve my app? Classifying user reviews for software maintenance and evolution. In *Proceedings of the IEEE International Conference on Software Maintenance and Evolution (ICSME'15)*. IEEE Computer Society, 281–290. DOI: https://doi.org/10.1109/ICSM.2015.7332474

[194] Sebastiano Panichella, Andrea Di Sorbo, Emitza Guzman, Corrado Aaron Visaggio, Gerardo Canfora, and Harald C. Gall. 2015. How can I improve my app? Classifying user reviews for software maintenance and evolution. In *Proceedings of the International Conference on Software Maintenance and Evolution*, Rainer Koschke, Jens Krinke, and Martin P. Robillard (Eds.). IEEE Computer Society, 281–290. DOI: https://doi.org/10.1109/ICSM.2015.7332474

[195] Sebastiano Panichella, Andrea Di Sorbo, Emitza Guzman, Corrado Aaron Visaggio, Gerardo Canfora, and Harald C. Gall. 2016. ARdoc: App reviews development oriented classifier. In *Proceedings of the 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, Thomas Zimmermann, Jane Cleland-Huang, and Zhendong Su (Eds.). ACM, 1023–1027. DOI: https://doi.org/10.1145/2950290.2983938

[196] Nikolaos Pappas and Andrei Popescu-Belis. 2013. Sentiment analysis of user comments for one-class collaborative filtering over TED talks. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'13)*. 773–776.

[197] Amol Patwardhan. 2017. Sentiment identification for collaborative, geographically dispersed, cross-functional software development teams. In *Proceedings of the 3rd IEEE International Conference on Collaboration and Internet Computing*. IEEE Computer Society, 20–26. DOI: https://doi.org/10.1109/CIC.2017.00014

[198] Rajshakhar Paul, Amiangshu Bosu, and Kazi Zakia Sultana. 2019. Expressions of sentiments during code reviews: Male vs. Female. In *Proceedings of the 26th IEEE International Conference on Software Analysis, Evolution and Reengineering*, Xinyu Wang, David Lo, and Emad Shihab (Eds.). IEEE, 26–37. DOI: https://doi.org/10.1109/SANER.2019.8667987

[199] Timo Pawelka and Elmar Jürgens. 2015. Is this code written in English? A study of the natural language of comments and identifiers in practice. In *Proceedings of the IEEE International Conference on Software Maintenance and Evolution*, Rainer Koschke, Jens Krinke, and Martin P. Robillard (Eds.). IEEE Computer Society, 401–410. DOI: https://doi.org/10.1109/ICSM.2015.7332491

[200] Zhenlian Peng, Jian Wang, Keqing He, and Mingdong Tang. 2016. An approach of extracting feature requests from app reviews. In *Proceedings of the 12th International Conference on Collaborate Computing: Networking, Applications and Worksharing (Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Vol. 201)*, Shangguang Wang and Ao Zhou (Eds.). Springer, 312–323. DOI: https://doi.org/10.1007/978-3-319-59288-6_28

[201] K. Phetrungnapha and T. Senivongse. 2019. Classification of mobile application user reviews for generating tickets on issue tracking system. In *Proceedings of the 12th International Conference on Information Communication Technology and System (ICTS)*. 229–234. DOI: https://doi.org/10.1109/ICTS.2019.8850962

[202] Daniel Pletea, Bogdan Vasilescu, and Alexander Serebrenik. 2014. Security and emotion: Sentiment analysis of security discussions on GitHub. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, Premkumar T. Devanbu, Sung Kim, and Martin Pinzger (Eds.). ACM, 348–351. DOI: https://doi.org/10.1145/2597073.2597117

[203] Roxana Lisette Quintanilla Portugal and Julio Cesar Sampaio do Prado Leite. 2018. Usability related qualities through sentiment analysis. In *Proceedings of the 1st International Workshop on Affective Computing for Requirements Engineering*, Davide Fucci, Nicole Novielli, and Emitza Guzman (Eds.). IEEE, 20–26. DOI: https://doi.org/10.1109/AffectRE.2018.00010

[204] Zhenzheng Qian, Beijun Shen, Wenkai Mo, and Yuting Chen. 2016. SatiIndicator: Leveraging user reviews to evaluate user satisfaction of sourceforge projects. In *Proceedings of the 40th IEEE Annual Computer Software and Applications Conference*. IEEE Computer Society, 93–102. DOI: https://doi.org/10.1109/COMPSAC.2016.183

[205] Zhenzheng Qian, Chengcheng Wan, and Yuting Chen. 2016. Evaluating quality-in-use of FLOSS through analyzing user reviews. In *Proceedings of the 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, Yihai Chen (Ed.). IEEE Computer Society, 547–552. DOI: https://doi.org/10.1109/SNPD.2016.7515956

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

[207] Kumar Ravi and Vadlamani Ravi. 2015. A survey on opinion mining and sentiment analysis: Tasks, approaches and applications. *Knowl.-based Syst.* 89 (2015), 14–46. DOI: https://doi.org/10.1016/j.knosys.2015.06.015

[208] Romain Robbes and Andrea Janes. 2019. Leveraging small software engineering data sets with pre-trained neural networks. In *Proceedings of the 41st International Conference on Software Engineering: New Ideas and Emerging Results.*, Anita Sarma and Leonardo Murta (Eds.). IEEE/ACM, 29–32. DOI: https://doi.org/10.1109/ICSE-NIER.2019.00016

[209] William N. Robinson, Tianjie Deng, and Zirun Qi. 2016. Developer behavior and sentiment from data mining open source repositories. In *Proceedings of the 49th Hawaii International Conference on System Sciences*, Tung X. Bui and Ralph H. Sprague Jr. (Eds.). IEEE Computer Society, 3729–3738. DOI: https://doi.org/10.1109/HICSS.2016.465

[210] Mary Sánchez-Gordón and Ricardo Colomo-Palacios. 2019. Taking the emotional pulse of software engineering—A systematic literature review of empirical studies. *Inf. Softw. Technol.* 115 (2019), 23–43.

[211] Mateus F. Santos, Josemar Alves Caetano, Johnatan Oliveira, and Humberto T. Marques Neto. 2018. Analyzing the impact of feedback in GitHub On the software developer's mood. In *Proceedings of the 30th International Conference on Software Engineering and Knowledge Engineering*, Óscar Mortágua Pereira (Ed.). KSI Research Inc. and Knowledge Systems Institute Graduate School, 445–444. DOI: https://doi.org/10.18293/SEKE2018-153

[212] Hitesh Sapkota, Pradeep K. Murukannaiah, and Yi Wang. 2020. A network-centric approach for estimating trust between open source software developers. *PLoS One* 14, 12 (12 2020), 1–30. DOI: https://doi.org/10.1371/journal.pone.0226281

[213] Simone Scalabrino, Gabriele Bavota, Barbara Russo, Massimiliano Di Penta, and Rocco Oliveto. 2019. Listening to the crowd for the release planning of mobile apps. *IEEE Trans. Softw. Eng.* 45, 1 (2019), 68–86. DOI: https://doi.org/10.1109/TSE.2017.2759112

[214] Simone Scalabrino, Gabriele Bavota, Barbara Russo, Massimiliano Di Penta, and Rocco Oliveto. 2019. Listening to the crowd for the release planning of mobile apps. *IEEE Trans. Softw. Eng.* 45, 1 (2019), 68–86. DOI: https://doi.org/10.1109/TSE.2017.2759112

[215] Ryan Serva, Zachary R. Senzer, Lori L. Pollock, and K. Vijay-Shanker. 2015. Automatically mining negative code examples from software developer Q & A forums. In *Proceedings of the 30th IEEE/ACM International Conference on Automated Software Engineering Workshops*. IEEE Computer Society, 115–122. DOI: https://doi.org/10.1109/ASEW.2015.10

[216] Faiz Ali Shah, Yevhenii Sabanin, and Dietmar Pfahl. 2016. Feature-based evaluation of competing apps. In *Proceedings of the International Workshop on App Market Analytics*, Meiyappan Nagappan, Federica Sarro, and Emad Shihab (Eds.). ACM, 15–21. DOI: https://doi.org/10.1145/2993259.2993267

[217] Faiz Ali Shah, Kairit Sirts, and Dietmar Pfahl. 2019. Using app reviews for competitive analysis: Tool support. In *Proceedings of the 3rd ACM SIGSOFT International Workshop on App Market Analytics*, Federica Sarro and Maleknaz Nayebi (Eds.). ACM, 40–46. DOI: https://doi.org/10.1145/3340496.3342756

[218] Jingyi Shen, Olga Baysal, and M. Omair Shafiq. 2019. Evaluating the performance of machine learning sentiment analysis algorithms in software engineering. In *Proceedings of the International Conference on Dependable, Autonomic and Secure Computing, International Conference on Pervasive Intelligence and Computing, International Conference on Cloud and Big Data Computing, International Conference on Cyber Science and Technology Congress*. IEEE, 1023–1030. DOI: https://doi.org/10.1109/DASC/PiCom/CBDCom/CyberSciTech.2019.00185

[219] Lin Shi, Celia Chen, Qing Wang, Shoubin Li, and Barry W. Boehm. 2017. Understanding feature requests by leveraging fuzzy method and linguistic analysis. In *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering*, Grigore Rosu, Massimiliano Di Penta, and Tien N. Nguyen (Eds.). IEEE Computer Society, 440–450. DOI: https://doi.org/10.1109/ASE.2017.8115656

[220] Navdeep Singh and Paramvir Singh. 2017. How do code refactoring activities impact software developers' sentiments?—An empirical investigation into GitHub commits. In *Proceedings of the 24th Asia-Pacific Software Engineering Conference*, Jian Lv, He Jason Zhang, Mike Hinchey, and Xiao Liu (Eds.). IEEE Computer Society, 648–653. DOI: https://doi.org/10.1109/APSEC.2017.79

[221] Vinayak Sinha, Alina Lazar, and Bonita Sharif. 2016. Analyzing developer sentiment in commit logs. In *Proceedings of the 13th International Conference on Mining Software Repositories (MSR'16)*. ACM, 520–523. DOI: https://doi.org/10.1145/2901739.2903501

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

[223] Ekaterina Skriptsova, Elizaveta Voronova, Elizaveta Danilova, and Alina Bakhitova. 2019. Analysis of newcomers activity in communicative posts on GitHub. In *Digital Transformation and Global Society*, Daniel A. Alexandrov, Alexander V. Boukhanovsky, Andrei V. Chugunov, Yury Kabanov, Olessia Koltsova, and Ilya Musabirov (Eds.). Springer International Publishing, Cham, 452–460.

[224] Tom De Smedt and Walter Daelemans. 2012. Pattern for Python. *J. Mach. Learn. Res.* 13 (2012), 2063–2067. Retrieved from http://dl.acm.org/citation.cfm?id=2343710.

[225] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'13)*. ACL, 1631–1642. Retrieved from https://www.aclweb.org/anthology/D13-1170/.

[226] Andrea Di Sorbo, Sebastiano Panichella, Carol V. Alexandru, Junji Shimagaki, Corrado Aaron Visaggio, Gerardo Canfora, and Harald C. Gall. 2016. What would users change in my app? Summarizing app reviews for recommending software changes. In *Proceedings of the 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, Thomas Zimmermann, Jane Cleland-Huang, and Zhendong Su (Eds.). ACM, 499–510. DOI: https://doi.org/10.1145/2950290.2950299

[227] Andrea Di Sorbo, Sebastiano Panichella, Corrado Aaron Visaggio, Massimiliano Di Penta, Gerardo Canfora, and Harald C. Gall. 2015. Development emails content analyzer: Intention mining in developer discussions. In *Proceedings of the 30th IEEE/ACM International Conference on Automated Software Engineering*, Myra B. Cohen, Lars Grunske, and Michael Whalen (Eds.). IEEE Computer Society, 12–23. DOI: https://doi.org/10.1109/ASE.2015.12

[228] Rodrigo R. G. Souza and Bruno Silva. 2017. Sentiment analysis of Travis CI builds. In *Proceedings of the 14th International Conference on Mining Software Repositories*, Jesús M. González-Barahona, Abram Hindle, and Lin Tan (Eds.). IEEE Computer Society, 459–462. DOI: https://doi.org/10.1109/MSR.2017.27

[229] Christoph Stanik, Marlo Häring, and Walid Maalej. 2019. Classifying multilingual user feedback using traditional machine learning and deep learning. In *Proceedings of the 27th IEEE International Requirements Engineering Conference Workshops*. IEEE, 220–226. DOI: https://doi.org/10.1109/REW.2019.00046

[230] Mohammadali Tavakoli, Liping Zhao, Atefeh Heydari, and Goran Nenadić. 2018. Extracting useful software development information from mobile application reviews: A survey of intelligent mining techniques and tools. *Exp. Syst. Applic.* 113 (2018), 186–199.

[231] Mike Thelwall. 2017. TensiStrength: Stress and relaxation magnitude detection for social media texts. *Inf. Process. Manag.* 53, 1 (2017), 106–121. DOI: https://doi.org/10.1016/j.ipm.2016.06.009

[232] Mike Thelwall, Kevan Buckley, Georgios Paltoglou, Di Cai, and Arvid Kappas. 2010. Sentiment strength detection in short informal text. *J. Assoc. Inf. Sci. Technol.* 61, 12 (2010), 2544–2558. DOI: https://doi.org/10.1002/asi.21416

[233] Parastou Tourani and Bram Adams. 2016. The impact of human discussions on just-in-time quality assurance: An empirical study on openstack and eclipse. In *Proceedings of the IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering*. IEEE Computer Society, 189–200. DOI: https://doi.org/10.1109/SANER.2016.113

[234] Parastou Tourani, Yujuan Jiang, and Bram Adams. 2014. Monitoring sentiment in open source mailing lists: Exploratory study on the Apache ecosystem. In *Proceedings of 24th Annual International Conference on Computer Science and Software Engineering*, Joanna Ng, Jin Li, and Ken Wong (Eds.). IBM/ACM, 34–44. Retrieved from http://dl.acm.org/citation.cfm?id=2735528.

[235] Andrew Truelove, Farah Naz Chowdhury, Omprakash Gnawali, and Mohammad Amin Alipour. 2019. Topics of concern: Identifying user issues in reviews of IoT apps and devices. In *Proceedings of the 1st International Workshop on Software Engineering Research & Practices for the Internet of Things*. IEEE/ACM, 33–40. DOI: https://doi.org/10.1109/SERP4IoT.2019.00013

[236] G. Uddin and F. Khomh. 2019. Automatic mining of opinions expressed about APIs in stack overflow. *IEEE Trans. Softw. Eng.* 122 (2019), 1–1. DOI: https://doi.org/10.1109/TSE.2019.2900245

[237] Gias Uddin, Foutse Khomh, and Chanchal K. Roy. 2020. Mining API usage scenarios from stack overflow. *Inf. Softw. Technol.* 122 (2020), 106277. DOI: https://doi.org/10.1016/j.infsof.2020.106277

[238] Lorenzo Villarroel, Gabriele Bavota, Barbara Russo, Rocco Oliveto, and Massimiliano Di Penta. 2016. Release planning of mobile apps based on user reviews. In *Proceedings of the 38th International Conference on Software Engineering (ICSE'16)*. ACM, 14–24. DOI: https://doi.org/10.1145/2884781.2884818

[239] Phong Minh Vu, Hung Viet Pham, Tam The Nguyen, and Tung Thanh Nguyen. 2015. Tool support for analyzing mobile app reviews. In *Proceedings of the 30th IEEE/ACM International Conference on Automated Software Engineering*, Myra B. Cohen, Lars Grunske, and Michael Whalen (Eds.). IEEE Computer Society, 789–794. DOI: https://doi.org/10.1109/ASE.2015.101

[240] Chong Wang, Maya Daneva, Marten van Sinderen, and Peng Liang. 2019. A systematic mapping study on crowdsourced requirements engineering using user feedback. *J. Softw.: Evol. Process* 31, 10 (2019), e2199.

[241] Shaohua Wang, NhatHai Phan, Yan Wang, and Yong Zhao. 2019. Extracting API tips from developer question and answer websites. In *Proceedings of the 16th International Conference on Mining Software Repositories*, Margaret-Anne D. Storey, Bram Adams, and Sonia Haiduc (Eds.). IEEE/ACM, 321–332. DOI: https://doi.org/10.1109/MSR.2019.00058

[242] Yi Wang. 2019. Emotions extracted from text vs. true emotions-an empirical evaluation in SE context. In *Proceedings of the 34th IEEE/ACM International Conference on Automated Software Engineering*. IEEE, 230–242. DOI: https://doi.org/10.1109/ASE.2019.00031

[243] Yue Wang, Hongning Wang, and Hui Fang. 2017. Extracting user-reported mobile application defects from online reviews. In *Proceedings of the IEEE International Conference on Data Mining Workshops*, Raju Gottumukkala, Xia Ning, Guozhu Dong, Vijay Raghavan, Srinivas Aluru, George Karypis, Lucio Miele, and Xindong Wu (Eds.). IEEE Computer Society, 422–429. DOI: https://doi.org/10.1109/ICDMW.2017.61

[244] Karl Werder. 2018. The evolution of emotional displays in open source software development teams: An individual growth curve analysis. In *Proceedings of the 3rd International Workshop on Emotion Awareness in Software Engineering*. ACM, 1–6. DOI: https://doi.org/10.1145/3194932.3194934

[245] Karl Werder. 2018. The evolution of emotional displays in open source software development teams: An individual growth curve analysis. In *Proceedings of the 3rd International Workshop on Emotion Awareness in Software Engineering*, Andrew Begel, Alexander Serebrenik, and Daniel Graziotin (Eds.). ACM, 1–6. DOI: https://doi.org/10.1145/3194932.3194934

[246] Karl Werder and Sjaak Brinkkemper. 2018. MEME: Toward a method for emotions extraction from GitHub. In *Proceedings of the 3rd International Workshop on Emotion Awareness in Software Engineering*, Andrew Begel, Alexander Serebrenik, and Daniel Graziotin (Eds.). ACM, 20–24. DOI: https://doi.org/10.1145/3194932.3194941

[247] Colin Werner, Ze Shi Li, and Daniela E. Damian. 2019. Can a machine learn through customer sentiment?: A cost-aware approach to predict support ticket escalations. *IEEE Softw.* 36, 5 (2019), 38–45. DOI: https://doi.org/10.1109/MS.2019.2923408

[248] Colin Werner, Ze Shi Li, and Neil A. Ernst. 2019. What can the sentiment of a software requirements specification document tell us? In *Proceedings of the 27th IEEE International Requirements Engineering Conference Workshops*. IEEE, 106–107. DOI: https://doi.org/10.1109/REW.2019.00022

[249] Colin Werner, Gabriel Tapuc, Lloyd Montgomery, Diksha Sharma, Sanja Dodos, and Daniela E. Damian. 2018. How angry are your customers? Sentiment analysis of support tickets that escalate. In *Proceedings of the 1st International Workshop on Affective Computing for Requirements Engineering*, Davide Fucci, Nicole Novielli, and Emitza Guzman (Eds.). IEEE, 1–8. DOI: https://doi.org/10.1109/AffectRE.2018.00006

[250] Grant Williams and Anas Mahmoud. 2017. Analyzing, classifying, and interpreting emotions in software users' tweets. In *Proceedings of the 2nd IEEE/ACM International Workshop on Emotion Awareness in Software Engineering*. IEEE Computer Society, 2–7. DOI: https://doi.org/10.1109/SEmotion.2017.1

[251] Grant Williams and Anas Mahmoud. 2017. Mining Twitter feeds for software user requirements. In *Proceedings of the 25th IEEE International Requirements Engineering Conference*, Ana Moreira, João Araújo, Jane Hayes, and Barbara Paech (Eds.). IEEE Computer Society, 1–10. DOI: https://doi.org/10.1109/RE.2017.14

[252] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in Software Engineering*. Springer Science & Business Media.

[253] Junfang Wu, Chunyang Ye, and Hui Zhou. 2021. BERT for sentiment classification in software engineering. In *Proceedings of the International Conference on Service Science (ICSS)*. 115–121. DOI: https://doi.org/10.1109/ICSS53362.2021.00026

[254] Bo Yang, Xinjie Wei, and Chao Liu. 2017. Sentiments analysis in GitHub repositories: An empirical study. In *Proceedings of the 24th Asia-Pacific Software Engineering Conference Workshops-*. IEEE, 84–89. DOI: https://doi.org/10.1109/APSECW.2017.13

[255] Huishi Yin and Dietmar Pfahl. 2017. A method to transform automatically extracted product features into inputs for Kano-like models. In *18th International Conference on Product-Focused Software Process Improvement (Lecture Notes in Computer Science, Vol. 10611)*, Michael Felderer, Daniel Méndez Fernández, Burak Turhan, Marcos Kalinowski, Federica Sarro, and Dietmar Winkler (Eds.). Springer, 237–254. DOI: https://doi.org/10.1007/978-3-319-69926-4_17

[256] Huishi Yin and Dietmar Pfahl. 2018. The OIRE method—Overview and initial validation. In *Proceedings of the 25th Asia-Pacific Software Engineering Conference*. IEEE, 1–10. DOI: https://doi.org/10.1109/APSEC.2018.00014

[257] Ting Zhang, Bowen Xu, Ferdian Thung, Stefanus Agus Haryono, David Lo, and Lingxiao Jiang. 2020. Sentiment analysis for software engineering: How far can pre-trained transformer models go? In *Proceedings of the IEEE International Conference on Software Maintenance and Evolution (ICSME)*. 70–80. DOI: https://doi.org/10.1109/ICSME46990.2020.00017

[258] Yingying Zhang and Daqing Hou. 2013. Extracting problematic API features from forum discussions. In *Proceedings of the IEEE 21st International Conference on Program Comprehension*. IEEE Computer Society, 142–151. DOI: https://doi.org/10.1109/ICPC.2013.6613842

[259] Lingling Zhao and Anping Zhao. 2019. Sentiment analysis based requirement evolution prediction. *Fut. Internet* 11, 2 (2019), 52. DOI: https://doi.org/10.3390/fi11020052

[260] Wayne Xin Zhao, Jing Jiang, Jianshu Weng, Jing He, Ee-Peng Lim, Hongfei Yan, and Xiaoming Li. 2011. Comparing Twitter and traditional media using topic models. In *Proceedings of the 33rd European Conference on IR Research on Advances in Information Retrieval (Lecture Notes in Computer Science, Vol. 6611)*. Springer, 338–349. DOI: https://doi.org/10.1007/978-3-642-20161-5_34

[261] Shenghua Zhou, S. Thomas Ng, Sang Hoon Lee, Frank J. Xu, and Yifan Yang. 2019. A domain knowledge incorporated text mining approach for capturing user needs on BIM applications. *Eng., Construct. Archit. Manag.* 27, 2 (2019).

[262] Yanzhen Zou, Changsheng Liu, Yong Jin, and Bing Xie. 2013. Assessing software quality through web comment search and analysis. In *13th International Conference on Software Reuse: Safe and Secure Software Reuse (Lecture Notes in Computer Science, Vol. 75)*, John M. Favaro and Maurizio Morisio (Eds.). Springer, 208–223. DOI: https://doi.org/10.1007/978-3-642-38977-1_14

Received March 2021; revised September 2021; accepted October 2021