Sentiment analysis for software quality assessment

Feng Liang[1], Fang Hou[1], Siamak Farshidi[1] and Slinger Jansen[1,2]

[1]*Utreht University, Utrecht, The Netherlands*
[2]*Lappeenranta University, Lappeenranta, Finland*

Abstract
During the software selection process, software engineers often rely on text reviews from repository platforms, communities, or forums to collect software quality information. As these reviews offer direct insight into users' experience and perception of the software components. However, text reviews are often formulated implicitly, and the process of gathering user feedback from multiple sources can be a time-intensive endeavor, posing challenges in the collection and analysis of substantial volumes of data. We conducted a systematic literature review to explore the state-of-the-art solutions in sentiment analysis. By leveraging the knowledge derived from the literature review, we developed a sentiment analysis tool to measure software component quality by analyzing the sentiment of reviews from experienced users. Our goal is to provide a channel to help software stakeholders gain insight into the software quality attributes, thus enhancing the overall health of software and the software ecosystem. This tool consists of TextRank, which extracts keywords related to software quality attributes from raw data, an Aho-Corasick automaton used to search for these keywords in reviews and map them to software quality attributes, and a sentiment analysis model to perform sentiment analysis. We compare four widely mentioned models in the literature review, namely BERT, BERT-BiLSTM, BERT-BiLSTM-Attention, and RoBERTa, in terms of performance metrics such as accuracy, F1, precision, and recall. BERT-BiLSTM-Attention is selected as the sentiment analysis model due to its superior performance in both training and test datasets. In addition, we integrated a decision algorithm that computes the fuzzy group consensus sentiment for the relevant quality attributes of each software component and visualizes it through a sentiment quality matrix.

Keywords
Sentiment analysis, software quality, software engineering

1. Introduction

Today's software end-users expect more from their software components than ever before. To speed up software development, shorten time-to-market, and reduce costs, software-producing organizations often integrate third-party software components into their products. Software development based on those third-party components is significantly dependent on the selection of reliable components. Our previous systematic literature review (SLR) and interviews with 24 software practitioners from various domains revealed that software practitioners typically collect measurable metrics, activity data, discussions, and text reviews from repository platforms,

---

*Feng Liang and Fang Hou are co-first authors, equally contributed to this paper.

BENEVOL'23: The 22nd Belgium-Netherlands Software Evolution Workshop Nijmegen, 27-28 November 2023

l.feng1@students.uu.nl (F. Liang); f.hou@uu.nl (F. Hou); s.farshidi@uu.nl (S. Farshidi); slinger.jansen@uu.nl (S. Jansen)
0000-0002-8042-3278 (F. Hou); 0000-0003-3270-4398 (S. Farshidi); 0000-0003-3752-2868 (S. Jansen)

communities, and forums to assess the trustworthiness or quality of software components [1, 2]. However, the process of aggregating text reviews from multiple online sources is a time-consuming task, and the inherent biases present in such reviews often fail to provide a precise representation of the broader user sentiment. Consequently, this can result in imprecise software quality assessments and pose challenges in reaching meaningful and reliable conclusions [3].

Sentiment can be defined as the opinions expressed by stakeholders [4], such as software end-users, software engineers, and consultants, regarding the quality of a software component. In software engineering (SE), sentiment analysis techniques are commonly used in opinion mining techniques to identify sentiments and subjective opinions expressed in text. For example, sentiment analysis is commonly used for analyzing commit comments in Github [5], discussion in Stack Overflow [6], as well as for evaluating software quality in conjunction with software quality models [7]. However, existing sentiment analysis tools specifically designed for SE have not achieved outstanding levels of accuracy [8]. Jongeling et al. [9] apply four sentiment analysis tools, including Sentistrength, NLTK, Stanford CoreNLP, and Alchemy API, on a text comment dataset obtained from JIRA, but none of them achieved unsatisfactory accuracy scores. The reasons could be:

- **Cannot capture and understand SE-related terms**: SE-related terms make a difference to the meaning and sentiment expressed in social events. For instance, "patch" and "commit" are technical terms and do not express sentiment [9]. Existing sentiment analysis models often struggle to effectively capture and comprehend specialized or domain-specific terminology in SE-related text documents. This limitation can hinder their ability to accurately grasp the sentiment and intended meaning conveyed within the content [6].
- **Cannot understand complex text**: subjective preferences or opinions may be expressed in the same comment, as well as objective technical information. Sentiment analysis models have difficulty understanding these distinctions [10].
- **Limited availability of labeled data sets**: Developing accurate sentiment analysis models requires access to labeled datasets specific to SE. However, there is a shortage of such datasets [6]. Models trained and evaluated on specific datasets can lead to bias towards certain sentiment labels, potentially skewing the assessment of software quality [11].

Based on the above context, we propose a tool to assess software quality by analyzing the sentiment of text reviews from websites. This tool assists software stakeholders in identifying the sentiments expressed in text reviews on a software component, as well as the related quality attributes. However, sentiments expressed across these diverse software quality attributes may be varied, encompassing positive, negative, or neutral sentiments. To address this variability, we employ a decision-making algorithm that calculates a fuzzy group consensus sentiment from the range of sentiments associated with each software quality attribute. This approach facilitates the clear presentation of the collective sentiment agreement for each attribute.

The remainder of the paper is organized as follows. Section 2 discusses the research method of this study. Section 3 briefly the conceptual model of this tool. Section 4 offers an overview of the experiment setting and the evaluation of candidate sentiment analysis models. Section 5 is the conclusion and future work.

2. Research Approach

The research questions of this study include the following:

- **MRQ:** *How to assess software quality by detecting sentiment expressed by stakeholders?*
- **RQ1:** *What approaches are available in the literature for detecting sentiments in text reviews?*
- **RQ2:** *What features do the sentiment analysis approach support?*
- **RQ3:** *Which aspects should be considered to measure the software quality?*
- **RQ4:** *How can an artifact be designed to assess the software quality from text reviews?*
- **RQ5:** *How can the artifact be evaluated?*

To answer RQ1, RQ2, and RQ3, we performed an SLR following the guidelines and steps of Kitchenham [12]. A set of 140 manuscripts since 2011 was identified. The complete SLR protocol and extraction results are available as [1]online material. In this study, we answer RQ4 by proposing a sentiment analysis tool.

3. Design Decisions

The purpose of this paper is to propose a tool for assessing the quality of software based on user reviews through sentiment analysis. It should include a feature for extracting keywords related to software quality, a feature for searching keywords from user reviews, and a model for sentiment analysis. The design decisions were made by using the knowledge collected during the SLR.

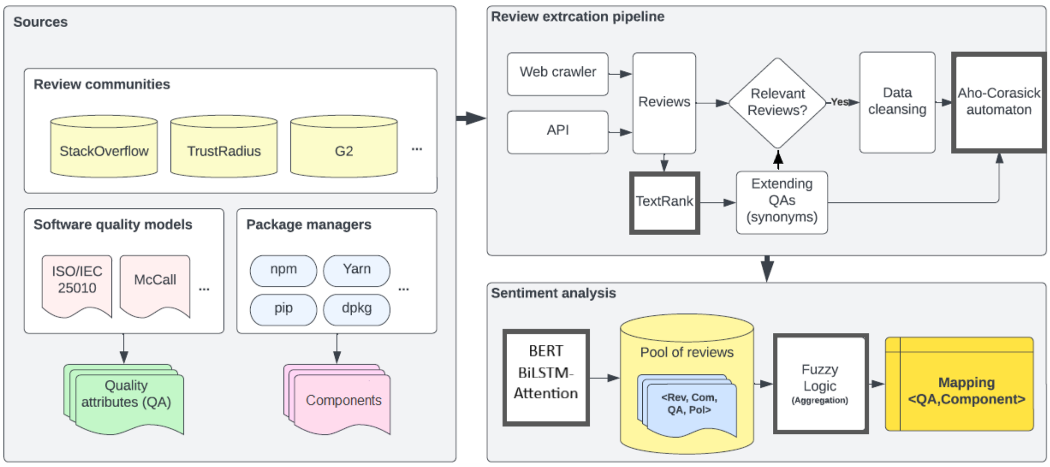

Figure 1: The structure of our sentiment analysis model, including data sources, review extraction pipeline, and sentiment analysis

[1]https://figshare.com/s/c04d123c1de8892287c6

3.1. Sources

The tool is designed to recognize and catalog software component names and their version from the user reviews. Based on Figure 1, sources include three parts. (1) Review communities: This is the source of review comments. Our approach involves the extraction of user reviews regarding software quality from online websites or communities. (2) Software quality and keyword mapping: We extracted keywords or keyphrases related to software quality from the review comments by TextRank, and created a mapping list in CSV format to map these keywords or phrases to relevant quality attributes manually. If possible, we will invite experts to check and confirm the mapping. Future researchers can easily add extended keywords to the provided CSV-formatted list using the same format. The tool can automatically convert the keyword-quality attribute mappings into a dictionary format for seamless integration with the model. (3) Software package: To enhance the dataset's quality and focus on relevance, the tool only analyzes specific software components (versions) from certain package managers, such as npm or pip. Employing regular expressions, we identify and extract data containing component names and version information from a predefined list of software components. This information is then recorded in a maintainable list of software components. Expanding the list of software components is as simple as adding their names to the original software component text file we have provided. The code can identify the software component names (versions) information within the input text.

3.2. Key-phrases extraction based on TextRank

Analyzing and extracting the keywords of quality attributes from user reviews is one of our needs. We are inspired by the solution proposed by Shuoqiu and Chaojun [13], Li and Shen [14]. They proposed a method of constructing a sentiment dictionary for online course reviews, TextRank has been adopted to extract the keywords from the reviews on websites. TextRank employs a graph-based ranking algorithm to extract keywords or key phrases from the input text [15]. It offers several advantages. First, it is an unsupervised algorithm, allowing for accurate key-phrase extraction even in the absence of a pre-annotated dataset [16]. Second, the TextRank algorithm demonstrates remarkable efficiency and high processing speed [15], making it suitable for handling substantial volumes of data.

3.3. Keyword matching based on Aho-Corasick

After the keyword extraction, we need to search them, including quality attributes keywords, and software packages (versions) from the user reviews. The combination of TextRank and Aho-Corasick has also been demonstrated to be a reasonable and effective combination in the literature. We draw on the research conducted by Li and Shen [14], Aho-Corasick automaton is used to conduct keyword matching on web pages, after the keywords generation by TextRank. When searching for multiple keywords at the same time in a large amount of text, the Aho-Corasick algorithm can find multiple occurrences of the keywords in the input text at once. Its time complexity is linear in the number of keywords appearing in the text, making it efficient to use even on large datasets [17].

3.4. Sentiment Analysis

The sentiment analysis model makes it possible to predict the sentiments of user reviews. These sentiments are assigned to the corresponding quality attributes. Based on the SLR results, we select four models that have been frequently adopted in sentiment analysis studies as the candidate models for sentiment analysis, namely BERT, BERT-BiLSTM, BERT-BiLSTM-Attention, and RoBERTa. The model with the best performance will be selected as the sentiment analysis model. To facilitate a comparative evaluation of these four models, we conducted the following experiment.

4. Experiment

In this section, we aim to evaluate the four models and select one as the sentiment analysis model.

4.1. Data sources

In this experiment, text reviews are sourced from several online communities, starting with Stack Overflow, TrustRadius, and G2, to encompass developer community platforms and user reviews of software. The sources are technical forum (Stack Overflow [2]), software review site (G2 [3]), and user feedback platform (TrustRadius [4]). Over 5,000,000 reviews have been extracted from the three sources as the raw dataset. To reflect current trends and patterns and incorporate new vocabularies and language usage, we only extracted user reviews from the last five years.

4.2. Data cleansing

A data cleaning process was performed, removing extraneous components that were not relevant to our analysis, such as URLs, HTML tags, code snippets, and special characters, as well as related software reviews for the components that do not exist in the software component list.

4.3. Data annotation

Annotated data is a critical prerequisite for conducting sentiment analysis. However, creating sentiment labels manually is a time-consuming task. Some studies directly employ gold datasets. We plan to explore an approach that ensures labeling accuracy while minimizing the need for extensive human effort. Wang et al. [18] conduct a study investigating the feasibility and performance of the GPT model for labeling tasks. Their results show that the use of GPT models can significantly reduce costs from 96% to 50% compared to manual annotators. First, we considered employing GPT to do the annotation. We performed stratified sampling on the training dataset, 4499 data were extracted to form the dataset. *GPT-turbo API* was adopted to perform sentiment analysis tasks with GPT-turbo models via API. The prompt used for the analysis is *"I want you to act like an expert in sentiment analysis in the software engineering*

[2]https://stackoverflow.com/
[3]https://www.g2.com/
[4]https://www.trustradius.com/

domain; can you please indicate the sentiment polarity of the given sentence? (Please say Negative, Neutral, or Positive)". Then two authors manually checked the results and we agreed with the annotation results for 3523 of the comments, with discrepancies in the results for the remaining 976. Only four were comments with opposite polarity (GPT annotated as positive or negative, and manually annotated as negative or positive). The manual annotation outcomes were employed as the labels for the respective data in data training and testing.

4.4. Model comparion

For the training process, we split 70% of the annotated data (3107 reviews) as training data to fine-tune models, and 30% for (1332 reviews) the final testing to measure the out-of-sample performance after training. Based on the SLR, the most frequently used performance indicators are Accuracy, F1, Precision, and Recall, the frequencies are 84, 73, 40, and 41. Table 1 shows the indicators for the four models. We can find that BERT-BiLSTM-Attention outweighs the other models.

	Training dataset				**Test dataset**			
	Accuracy	F1	Precision	Recall	Accuracy	F1	Precision	Recall
BERT	0.8979	0.8944	0.8933	0.8985	0.8033	0.7933	0.79433	0.8012
BERT-BiLSTM	0.9172	0.9141	0.9139	0.9157	0.8108	0.8001	0.8017	0.8028
BERT-BiLSTM-Attention	**0.9539**	**0.9523**	**0.9511**	**0.9546**	0.**8209**	**0.8139**	**0.8121**	**0.8187**
RoBERTa	0.8776	0.8709	0.8739	0.8745	0.8168	0.8043	0.8099	0.8126

Table 1
Evaluation results for sentiment analysis.

4.5. Fuzzy logical calculation

The sentiment analysis of software component quality attributes is often distributed, which may not readily convey whether an attribute is positive, negative, or neutral. Taking inspiration from Farshidi et al. [19], we consider it as a decision-making problem and solve it by applying the fuzzy logical calculation method proposed by Hsu and Chen [20]. The aim is to quantify the similarity of each pair of sentiments and to aggregate fuzzy individual sentiments into fuzzy group consensus sentiments. Figure 2 shows a partial result. In each cell, the first row contains the number of negative, neutral, and positive polarity. The decimal numbers in the second row are the fuzzy group consensus sentiment of this attribute. In the table, the intensity of the green color represents the strength of the "positive" polarity, while the intensity of the red color represents the strength of the "negative" polarity. Yellow color indicates "neutral" polarity.

5. Conclusions

In this paper, we present a solution for the task of sentiment analysis in the software engineering (SE) domain. This tool uses a TextRank model to extract keywords from the dataset, maps them to ISO/IEC 25010 software quality attributes by Aho-Corasick automaton, and utilizes BERT-BiLSTM-Attention to analyze the sentiment of software quality-related text reviews.

<table>
<tr><th rowspan="2">Software package</th><th colspan="18">Software Quality Attributes</th></tr>
<tr><th>Functional suitability</th><th>Functional correctness</th><th>Time behaviour</th><th>Resource utilization</th><th>Capacity</th><th>Compatibility</th><th>Co-existence</th><th>Interoperability</th><th>Usability</th><th>Reliability</th><th>Maturity</th><th>Availability</th><th>Fault tolerance</th><th>Recoverability</th><th>Security</th><th>Confidentiality</th><th>Integrity</th><th>Authenticity</th></tr>
<tr><td>opengl</td><td>6|5|5
0.48</td><td>7|5|5
0.46</td><td>17|26|11
0.46</td><td>48|97|29
0.46</td><td>1|1|1
0.5</td><td>39|85|21
0.46</td><td>12|33|2
0.42</td><td>39|85|21
0.46</td><td>44|100|31
0.47</td><td>9|2|4
0.38</td><td>0|0|3
0.86</td><td>8|1|1
0.25</td><td>15|5|8
0.41</td><td>1|1|0
0.32</td><td>0|2|1
0.62</td><td>0|1|1
0.68</td><td>1|3|1
0.5</td><td>0|1|0
0.5</td></tr>
<tr><td>nginx</td><td>16|29|22
0.53</td><td>18|31|22
0.52</td><td>45|115|42
0.49</td><td>84|351|90
0.5</td><td>7|25|6
0.49</td><td>70|326|74
0.5</td><td>3|7|5
0.55</td><td>70|326|74
0.5</td><td>84|405|100
0.51</td><td>16|7|4
0.34</td><td>5|5|7
0.54</td><td>14|15|2
0.36</td><td>28|34|27
0.5</td><td>2|13|0
0.45</td><td>22|86|12
0.47</td><td>12|41|3
0.44</td><td>0|19|5
0.58</td><td>11|44|7
0.48</td></tr>
<tr><td>django</td><td>21|32|10
0.44</td><td>22|34|12
0.45</td><td>49|97|44
0.49</td><td>154|691|195
0.51</td><td>2|4|7
0.64</td><td>143|689|177
0.51</td><td>35|82|29
0.49</td><td>143|689|177
0.51</td><td>175|813|263
0.53</td><td>19|9|9
0.4</td><td>4|13|13
0.61</td><td>14|7|4
0.36</td><td>32|35|18
0.44</td><td>2|17|4
0.53</td><td>58|269|45
0.49</td><td>12|20|3
0.41</td><td>22|51|28
0.52</td><td>50|251|42
0.49</td></tr>
<tr><td>apache-kafka</td><td>2|5|5
0.59</td><td>2|5|5
0.59</td><td>12|22|39
0.63</td><td>15|24|43
0.62</td><td>2|9|20
0.71</td><td>14|20|41
0.63</td><td>3|7|12
0.65</td><td>14|20|41
0.63</td><td>20|25|70
0.66</td><td>2|1|18
0.77</td><td>2|1|22
0.79</td><td>1|1|2
0.59</td><td>9|8|15
0.57</td><td>1|2|3
0.62</td><td>0|1|2
0.74</td><td></td><td>1|6|9
0.68</td><td>0|1|0
0.5</td></tr>
<tr><td>sqlite</td><td>4|1|12
0.67</td><td>4|1|12
0.67</td><td>11|6|26
0.63</td><td>20|15|58
0.65</td><td>10|4|5
0.41</td><td>20|15|47
0.62</td><td>6|4|7
0.52</td><td>20|15|47
0.62</td><td>24|14|183
0.76</td><td>4|3|10
0.63</td><td>3|4|20
0.73</td><td>5|1|5
0.5</td><td>9|4|23
0.64</td><td>0|2|3
0.72</td><td>5|2|1
0.32</td><td>4|2|1
0.35</td><td>1|3|20
0.79</td><td>1|0|0
0.14</td></tr>
<tr><td>redis</td><td>0|3|17
0.81</td><td>0|3|17
0.81</td><td>5|11|26
0.68</td><td>10|12|37
0.66</td><td>0|2|3
0.72</td><td>9|8|25
0.64</td><td>2|0|1
0.38</td><td>9|8|25
0.64</td><td>13|12|47
0.67</td><td>2|2|10
0.71</td><td>2|1|9
0.71</td><td>0|1|3
0.77</td><td>0|4|22
0.8</td><td>1|0|3
0.68</td><td>0|2|2
0.68</td><td></td><td>1|0|3
0.68</td><td>0|2|2
0.68</td></tr>
<tr><td>rapidminer-studio</td><td>5|5|11
0.6</td><td>5|5|11
0.6</td><td>17|22|51
0.64</td><td>37|41|116
0.65</td><td>9|6|10
0.51</td><td>30|30|102
0.66</td><td>8|6|8
0.5</td><td>30|30|102
0.66</td><td>54|64|314
0.72</td><td>3|0|6
0.62</td><td>2|2|11
0.72</td><td>4|4|6
0.55</td><td>9|6|17
0.59</td><td>0|0|1
0.86</td><td>1|0|0
0.14</td><td>1|0|0
0.14</td><td>1|2|8
0.73</td><td>1|0|0
0.14</td></tr>
</table>

Figure 2: A partial subset of the quality sentiment matrix regarding the attribute sentiment of the software packages from PIP.

Furthermore, our approach employs a decision-making algorithm to calculate the fuzzy group consensus sentiment for each attribute of software components. This facilitates a comprehensive and direct understanding of the quality attributes expressed in text reviews, along with their corresponding sentiments. Consequently, software stakeholders can easily grasp insights into software quality and optimize their software selection process based on this information.

Our future work will focus on the following aspects. First, we would like to work with more expert teams to customize and evaluate the keyword mapping to quality attributes. Second, we will explore a data annotation method that combines manual annotation and sophisticated AI models to improve annotation efficiency and achieve higher quality and larger capacity of the golden dataset. Third, we will improve the model, for instance, utilizing structured information from large-scale software engineering knowledge bases to improve its accuracy for sentiment analysis in software quality. Finally, we will integrate this tool into some platforms, for instance, TrustSECO [21], a community-managed infrastructure that we are developing that underpins the SECO with a trust layer.

References

[1] F. Hou, S. Jansen, A systematic literature review on trust in the software ecosystem, Empirical Software Engineering 28 (2023) 8.

[2] F. Hou, F. Jansen, A. de Vries, S. Jansen, The role of software trust in selection of open-source and closed software (2023) 30–37.

[3] G. Uddin, F. Khomh, Automatic mining of opinions expressed about apis in stack overflow, IEEE Transactions on Software Engineering 47 (2019) 522–559.

[4] R. Dehkharghani, C. Yilmaz, Automatically identifying a software product's quality attributes through sentiment analysis of tweets, in: 2013 1st International Workshop on Natural Language Analysis in Software Engineering (NaturaLiSE), IEEE, 2013, pp. 25–30.

[5] A. H. Ahmed Abbasi, M. Dhar, Benchmarking twitter sentiment analysis tools, in: Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14), Reykjavik, Iceland, may. European Language Resources Association (ELRA), 2014.

[6] B. Lin, F. Zampetti, G. Bavota, M. Di Penta, M. Lanza, R. Oliveto, Sentiment analysis for

software engineering: How far can we go?, in: Proceedings of the 40th international conference on software engineering, 2018, pp. 94–104.

[7] W. Leopairote, A. Surarerks, N. Prompoon, Evaluating software quality in use using user reviews mining, in: The 2013 10th International Joint Conference on Computer Science and Software Engineering (JCSSE), IEEE, 2013, pp. 257–262.

[8] J. Ding, H. Sun, X. Wang, X. Liu, Entity-level sentiment analysis of issue comments, in: 2018 IEEE/ACM 3rd International Workshop on Emotion Awareness in Software Engineering (SEmotion), 2018, pp. 7–13.

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

[10] W. Medhat, A. Hassan, H. Korashy, Sentiment analysis algorithms and applications: A survey, Ain Shams engineering journal 5 (2014) 1093–1113.

[11] M. Wankhade, A. C. S. Rao, C. Kulkarni, A survey on sentiment analysis methods, applications, and challenges, Artificial Intelligence Review 55 (2022) 5731–5780.

[12] B. Kitchenham, Procedures for performing systematic reviews, Keele, UK, Keele University 33 (2004) 1–26.

[13] Y. Shuoqiu, X. Chaojun, Research on constructing sentiment dictionary of online course reviews based on multi-source combination, in: Proceedings of the 2019 2nd International Conference on Data Science and Information Technology, 2019, pp. 71–76.

[14] Z. Li, Z. Shen, Deep semantic mining of big multimedia data advertisements based on needs ontology construction, Multimedia Tools and Applications 81 (2022) 28079–28102.

[15] R. Mihalcea, P. Tarau, Textrank: Bringing order into text, in: Proceedings of the 2004 conference on empirical methods in natural language processing, 2004, pp. 404–411.

[16] Y. Tao, Z. Cui, Z. Jiazhe, Research on keyword extraction algorithm using pmi and textrank, in: 2019 IEEE 2nd International Conference on Information and Computer Technologies (ICICT), IEEE, 2019, pp. 5–9.

[17] D. Pao, W. Lin, B. Liu, A memory-efficient pipelined implementation of the aho-corasick string-matching algorithm, ACM Transactions on Architecture and Code Optimization (TACO) 7 (2010) 1–27.

[18] S. Wang, Y. Liu, Y. Xu, C. Zhu, M. Zeng, Want to reduce labeling cost? gpt-3 can help, in: Findings of the Association for Computational Linguistics: EMNLP 2021, Association for Computational Linguistics, Punta Cana, Dominican Republic, 2021.

[19] S. Farshidi, S. Jansen, J. M. van der Werf, Capturing software architecture knowledge for pattern-driven design, Journal of Systems and Software 169 (2020) 110714.

[20] H.-M. Hsu, C.-T. Chen, Aggregation of fuzzy opinions under group decision making, Fuzzy sets and systems 79 (1996) 279–285.

[21] F. Hou, S. Farshidi, S. Jansen, Trustseco: A distributed infrastructure for providing trust in the software ecosystem, in: International Conference on Advanced Information Systems Engineering, Springer, 2021, pp. 121–133.