Large Language Models for Issue Report Classification

Giuseppe Colavito[1], Filippo Lanubile[1], Nicole Novielli[1] and Luigi Quaranta[1]

[1]University of Bari "Aldo Moro", Italy

Abstract

Effective issue classification is crucial for efficient software project management. However, labels assigned to issues are often inconsistent, which can negatively impact the performance of supervised classification models. In this work, we investigate how label consistency and training data size affect automatic issue classification. We first evaluate a few-shot learning approach on a manually validated dataset and compare it to fine-tuning on a larger crowd-sourced set. The results show that our approach achieves higher accuracy when trained and tested on consistent labels. We then examine zero-shot classification using GPT-3.5, finding that its performance is comparable to supervised models despite having no fine-tuning. This suggests that generative models can help classify issues when annotated data is limited. Overall, our findings provide insights into balancing data quantity and quality for issue classification.

Keywords

Issue classification, Large Language Models, Generative AI, Software Maintenance and Evolution, Few-Shot Learning

1. Introduction

Collaborative software development involves complex processes and activities to effectively support software development and maintenance. In this context, issue-tracking systems are widely adopted to manage requests for changes – such as bug fixes or product enhancements, as well as requests for support from users – and are regarded as essential tools for maintainers to efficiently manage software evolution activities.

Issue reports organized in such systems typically contain information such as an identifier, a description, the author, the issue status (e.g., open, assigned, closed), a comment thread, and a label indicating the type of issue, such as *bug*, *enhancement*, or *support*. Effective labeling of issue reports is of paramount importance to support prioritization and decision-making. Unfortunately, however, label misuse is a common problem, as submitters often confuse improvement requests as bugs and vice versa [1]. For example, Herzig et.al [2] reported that approximately 33.8% of all issue reports are incorrectly labeled. To avoid dealing with incorrect labels, automated classification methods have been proposed. Automatic issue classification can enable effective issue management and prioritization [3], without the need to instruct developers on how to assign labels correctly.

Early research on this topic proposed exploiting supervised methods that leverage text-based features for the task of automatic issue report classification [1]. More recently, approaches leveraging word embeddings have emerged [4, 5, 6, 7]. In particular, approaches based on BERT [8] and its variants achieved state-of-the-art performance [9, 10, 11].

In our previous work, we conducted an empirical study to investigate to what extent we can leverage pre-trained language models for automatic issue labeling [10]. We experimented with a dataset of more than 800K issue reports from GitHub open-source software projects labeled by project contributors as *bug*, *enhancement*, or *question* [9]. We fine-tuned the BERT [8] variant RoBERTa [12], achieving state-of-the-art performance (F1 = 0.8591).

Our manual error analysis revealed that the main cause of the misclassification of issues is label inconsistency across different projects. Also, several issue reports in the dataset were tagged with more than one label, which is indeed a source of noise. This evidence is in line with previous studies reporting the impact of data quality on the performance of machine learning models [13]. Informed by the results of our error analysis and by findings of previous research, we formulate the following research question:

RQ1: To what extent does label consistency impact the performance of supervised issue classification models?

To address it, we investigate the efficacy of few-shot learning for training robust classifiers using a small training dataset with manually validated labels. Specifically, we experiment with SETFIT, an effective methodology for fine-tuning of transformer-based models using few-shot learning [14], achieving promising results [15].

Still, manual annotation can be a costly task, both in terms of time and resources, even if done on a small set of manually curated examples. Hence, the need for minimizing the effort associated with data labeling re-

Ital-IA 2024: 4th National Conference on Artificial Intelligence, organized by CINI, May 29-30, 2024, Naples, Italy

giuseppe.colavito@uniba.it (G. Colavito); filippo.lanubile@uniba.it (F. Lanubile); nicole.novielli@uniba.it (N. Novielli); luigi.quaranta@uniba.it (L. Quaranta)

0000-0003-3871-401X (G. Colavito); 0000-0003-3373-7589 (F. Lanubile); 0000-0003-1160-2608 (N. Novielli); 0000-0002-9221-0739 (L. Quaranta)

mains. With the advent of recent GPT-like Large Language Models (LLMs), researchers have started investigating their potential in solving software engineering challenges [16, 17]. To better understand how GPT-like LLMs can be leveraged in automated issue labeling in the absence of training data, we formulate and investigate our second research question as follows:

RQ2: To what extent we can leverage GPT-like LLMs to classify issue reports?

To address it, we evaluate GPT3.5-turbo [18] in a zero-shot learning scenario, where the model is prompted by only providing the task and label descriptions. We compare the performance of classifiers based on GPT-like LLMs with fine-tuned BERT-like LLMs [19].

In this paper, we discuss our ongoing work on using LLMs to address software engineering challenges, with a particular focus on the automatic classification of issue reports in a low-resource setting. Specifically, we summarize the findings of two recent studies in which we addressed the research questions formulated above [15, 19]. The remainder of the paper is organized as follows. In Sections 2 and 3, we describe the datasets and methodology adopted in our empirical studies, respectively. Then, we report and discuss the study results in Section 4. The paper is concluded in Section 5, where we also outline directions for future work.

2. Dataset

To address our research questions, we use a dataset of 400 GitHub issues labeled as *bug*, *features*, *question*, and *documentation*. The dataset is split into two subsets of 200 issues which we use as train and test sets, respectively. Both subsets are equally distributed and include 50 issues per class. Our dataset is obtained by manually labeling the 400 randomly selected items from the dataset of 1.4M GitHub issues distributed by the NLBSE'23 tool competition organizers [20]. To manually ensure the consistency of labels in our dataset, three annotators individually categorized each issue report based on the information in its title and body. Each issue report was assigned to two of the annotators. We observed a Cohen's κ of 0.74, which indicates a substantial level of interrater agreement [21]. The annotators had a joint plenary meeting to discuss and resolve the cases of disagreement. Through this procedure, we ensured the reliability and consistency of the annotations. Table 1 presents the dataset's distribution before and after the manual labeling. The manually annotated sample is publicly available [22].

3. Methodology

To address our first research question, we investigate the efficacy of few-shot learning for training robust classifiers using the small manually validated training dataset described in Section 2. In particular, we train and evaluate a model based on SETFIT [14] using the manually labeled train and test sets. Then we compare its performance with the one obtained by fine-tuning RoBERTa [15] using the full dataset of 1.4M crowd-annotated issues [20].

Table 1
Distribution of labels in the extracted samples.

Label	Train set		Test set	
Bug	47	24%	53	27%
Documentation	33	17%	32	16%
Feature	60	30%	55	28%
Question	44	22%	47	24%
Discarded	16	8%	13	7%
Total	200		200	

To address our second research question, we compare the performance of the SETFIT classifier with the performance achieved by GPT 3.5 in a zero-shot learning scenario. We highlight that prompting is only used for GPT while the SETFIT model is trained on the manually labeled data. Both models are evaluated on the test set partition of manually labeled issues.

Preprocessing For our SETFIT model, we preprocess our dataset as follows. First, non-textual items, such as links, code snippets, and images, are identified and replaced with tokens (e.g., <link> for links) in the dataset. Next, we use the ekphrasis Text Pre-Processor [1] to normalize the text by detecting and replacing items such as URLs, email addresses, symbols, phone numbers, mentions, time, date, and numbers with specific tokens.

Choice of GPT-like models Several LLMs have been proposed in the last few years, with GPT-3 [23] being one of the most popular. There is a significant prevalence of studies leveraging GPT3.5-turbo [24], an instruction-tuned version of GPT-3, which is able to interact as a chatbot. For this reason, we select GPT3.5-turbo [18] as representative of GPT-like LLMs. We experiment with several versions of GPT3.5-turbo, with varying context length and date of training. Here we only report the results of the model with the best performance. More details can be found in our original work describing this study [19].

Prompting To instruct the model to perform the classification task, we create a prompt that includes the following items:

- *Input Format*: The format of the input issues, which includes a title and a body;

[1] https://github.com/cbaziotis/ekphrasis

- *Task Description*: A description of the classification task to be performed, including the possible labels that can be assigned to the issues;
- *Label Descriptions*: A brief description of each label. Label descriptions are generated by ChatGPT and then manually reviewed to ensure they are clear and informative.
- *Input Issue*: The issue to be classified;
- *Output format instructions*: The desired output format. We ask the model for a JSON object containing a reasoning and the predicted label. This is done to inject some Chain-of-Thought reasoning into the model, as suggested in previous studies about prompting LLMs [25, 26]. However, the reasoning serves as a prompt-engineering strategy and is not used to evaluate the model.

Evaluation In line with previous work [6, 7, 11, 10], the evaluation of the classifiers on the test set is provided in terms of precision, recall, and f1-measure [15]. For GPT-like LLMs, we parse the JSON response and extract the predicted label. In cases in which the label is not valid or the model did not follow the instructions appropriately, we discard the prediction. This process is done with the use of regular expressions. Both the models are tested on the manually verified test set [19].

4. Results and Discussion

4.1. Impact of label consistency on the classifier performance (RQ1)

In Table 2, we present the results obtained by training the SETFIT classifier on the hand-labeled gold standard and evaluating it on both the hand-labeled test set (a) and the full test set distributed for the challenge (c). To ensure a fair comparison, we compared the SETFIT model's performance with the performance obtained by RoBERTa on the same test set, when trained on the hand-labeled gold standard set (b1). Furthermore, we also include the performance obtained by training the RoBERTa classifier on the full train set distributed by the organizers (b2).

To assess the ability of the models to generalize on a broader dataset, we also include a comparison with the NLBSE '23 challenge baseline [20] (see row (d) of the table) and the SETFIT model's performance on the challenge full test set (see model (c) in the table). It is worth noting that the SETFIT model is designed to learn from a few examples. As such, it was not possible to train it on the raw dataset, since it is not optimized for such a setting and it would have been heavily time expensive. Instead, the RoBERTa baseline is trained on the full set.

The SETFIT model achieved an F1-micro score of .7767 (see model (c) in Table 2) when trained on the manually labeled gold standard and tested on the raw test set. When trained and evaluated on the manually labeled dataset (a), SETFIT performs better than RoBERTa (b1 and b2), regardless of whether the training set used for RoBERTa is raw or manually labeled. However, when trained on the manually-labeled dataset (b1), RoBERTa struggles to deliver good performance due to a shortage of training data. On the other hand, when trained on the raw dataset (b2), RoBERTa achieves competitive performances, but it is unable to outperform SETFIT (b).

As the manually-labeled dataset embodies the ideal labeling criteria for classifiers, comparing SETFIT (a) and RoBERTa (b2) provides a practical scenario in which we must choose either training a classifier on a large volume of data with disregard for data quality or concentrating on a smaller portion of data and manually improving label quality. This comparison suggests that data quality might be crucial for ensuring classification accuracy. A potential approach could be to start with a few-shot classifier and gradually switch to a more powerful model like RoBERTa when a fair amount of manually verified data becomes available. By doing so, we can strike a balance between data quantity and quality, ensuring that the classifier performs effectively while minimizing the possibility of inaccurate results caused by inconsistency in the labeling.

4.2. Leveraging GPT for automatic issue report classification (RQ2)

In Table 3, we report the classification performance of GPT compared to the SETFIT model. As already explained in the previous section, we experimented with several versions of GPT 3.5 that were available at the time of the study. For a full report of the results, see Colavito et al. [19]. In this paper, we include consideration of the 16k-0613 model only as this achieves the best performance in terms of a combination of F1 and percentage of discarded items due to nonsensical model output. Specifically, none of the predictions from this model were discarded. We observe that the Feature class achieves the best F1, while the Documentation class is the most problematic to identify, showing a lower recall than the other classes.

While the zero-shot GPT model achieves a slightly lower performance (F1 = .8155) than SETFIT (F1 = .8321), the models are still comparable. It's worth noting that SETFIT was fine-tuned on a portion of the issue report gold standard dataset, while GPT was evaluated in a zero-shot setting without any task-specific fine-tuning. This implies that GPT is capable of classifying issue reports with only a minor decrease in accuracy compared to fine-tuned BERT-like models. This presents a major benefit of GPT for this application since it can perform the classification in absence of labeled data, i.e., without the need

Table 2
Performance of the SETFIT model and comparison with the RoBERTa baseline approach. The performance of the model submitted to the challenge is reported in *Italic*. In bold, we highlight the best performance obtained with SETFIT.

	Model	Train		Test		F1
(a)	SETFIT	Sampled	Manual labels	Sampled	Manual labels	**0.8321**
(b1)	RoBERTa	Sampled	Manual labels	Sampled	Manual labels	0.4348
(b2)	RoBERTa	Full	GitHub labels	Sampled	Manual labels	0.8182
(c)	*SETFIT*	*Sampled*	*Manual labels*	*Full*	*GitHub labeling*	*0.7767*
(d)	RoBERTa (baseline)	Full	GitHub labels	Full	GitHub labels	0.8890

Table 3
Comparison between SETFIT and GPT-3.5.

	SETFIT			GPT-3.5 (16k-0613), zero-shot		
Label	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Bug	0.8723	0.8472	**0.8590**	0,7133	0,9811	0,8261
Documentation	0.9039	0.6594	**0.7616**	0,8853	0,6191	0,7285
Feature	0.7494	0.9182	0.8251	0,8861	0,8491	**0,8672**
Question	0.8754	0.8319	**0.8528**	0,8668	0,7719	0,8164
Overall	0.8321	0.8321	**0.8321**	0,8155	0,8155	0,8155

for fine-tuning. This evidence could help maintainers of new projects, for which historical data is not available or is scarce. In such cases, API calls to GPT could be used to classify issue reports, providing a valuable tool for project management. Once the project has accumulated enough labeled data, the maintainer could switch to a fine-tuned model to improve the classification accuracy. Although this could be a viable solution for open-source projects, it is worth noting that the cost of API calls and the privacy of data could limit its practical feasibility in commercial projects. In such cases, project maintainers might consider using open-source models or building and deploying a classifier on-premise. Nonetheless, the construction and maintenance of LLMs is expensive both in terms of resources and time, and this constitutes a barrier to their adoption in most cases.

5. Conclusion and Future Works

In this paper, we summarized the outcomes of our recently published studies on the use of large language models for automated issue classification. Specifically, we investigated the impact of improving data quality on issue classification performance. We trained and evaluated a model based on few-shot learning using SETFIT with a subset of manually verified data. The model achieves better performance when trained and tested on data for which label consistency was manually verified [22], compared to the RoBERTa baseline. However, RoBERTa generalizes better on the full test dataset when fine-tuned on the full crowd-sourced dataset.

Furthermore, we explored the performance of GPT-like models for automatic issue classification [19] to understand if we can leverage GPT-like LLMs to achieve state-of-the-art performance in the absence of manually annotated issues, i.e. when a gold standard is not available for fine-tuning state-of-the-art approaches based on BERT-like models. Our empirical results show that GPT-like models can achieve a performance comparable to the state-of-the-art without the need for fine-tuning. This suggests that when manual annotation is not feasible or a gold standard for training is not available (i.e., on a new project), maintainers could rely on generative AI to successfully address the issue classification task.

However, using LLMs to build issue classifiers might pose important challenges due to licensing and computational limitations. As such, we plan to extend this benchmark with open-source LLMs, also including issue-report datasets. This will enable evaluating the generalizability of our findings.

Acknowledgments

This research was co-funded by the NRRP Initiative, Mission 4, Component 2, Investment 1.3 - Partnerships extended to universities, research centres, companies and research D.D. MUR n. 341 del 15.03.2022 – Next Generation EU ("FAIR - Future Artificial Intelligence Research", code PE00000013, CUP H97G22000210007) and by the European Union - NextGenerationEU through the Italian Ministry of University and Research, Projects PRIN 2022 ("QualAI: Continuous Quality Improvement of AI-based Systems", grant n. 2022B3BP5S, CUP: H53D23003510006).

References

[1] G. Antoniol, K. Ayari, M. Di Penta, F. Khomh, Y.-G. Guéhéneuc, Is it a bug or an enhancement? a text-based approach to classify change requests, in: Proc. of the 2008 Conf. of the Center for Advanced Studies on Collaborative Research: Meeting of Minds, CASCON '08, ACM, New York, NY, USA, 2008. doi:10.1145/1463788.1463819.

[2] K. Herzig, S. Just, A. Zeller, It's not a bug, it's a feature: How misclassification impacts bug prediction, in: 2013 35th Int'l Conf.on Software Engineering (ICSE), 2013. doi:10.1109/ICSE.2013.6606585.

[3] N. Pandey, D. Sanyal, A. Hudait, A. Sen, Automated classification of software issue reports using machine learning techniques: an empirical study, Innovations in Systems and Software Engineering (2017). doi:10.1007/s11334-017-0294-1.

[4] O. Levy, Y. Goldberg, Neural word embedding as implicit matrix factorization, in: Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, K. Q. Weinberger (Eds.), Advances in Neural Information Processing Systems, Curran Assoc., Inc., 2014.

[5] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Proc. of the 26th Int'l Conf.on Neural Inf. Proc. Systems - Volume 2, NIPS'13, Curran Associates Inc., Red Hook, NY, USA, 2013.

[6] R. Kallis, A. Di Sorbo, G. Canfora, S. Panichella, Predicting issue types on github, Science of Computer Programming (2021). doi:https://doi.org/10.1016/j.scico.2020.102598.

[7] R. Kallis, A. Di Sorbo, G. Canfora, S. Panichella, Ticket tagger: Machine learning driven issue classification, in: 2019 IEEE Int'l. Conf on Software Maintenance and Evolution (ICSME), IEEE, 2019. doi:10.1109/ICSME.2019.00070.

[8] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in: Proc. of the 2019 Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, ACL, 2019. doi:10.18653/v1/N19-1423.

[9] R. Kallis, O. Chaparro, A. Di Sorbo, S. Panichella, Nlbse'22 tool competition, in: Proc. of The 1st Int'l Work. on Natural Language-based Software Eng. (NLBSE'22), 2022.

[10] G. Colavito, F. Lanubile, N. Novielli, Issue report classification using pre-trained language models, in: 2022 IEEE/ACM 1st Int'l Workshop on Natural Language-Based Software Eng. (NLBSE), IEEE Computer Society, USA, 2022. doi:10.1145/3528588.3528659.

[11] M. Izadi, CatIss: An Intelligent Tool for Categorizing Issues Reports using Transformers, in: (NLBSE 2022), 2022. doi:10.1145/3528588.3528662.

[12] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, Roberta: A robustly optimized bert pretraining approach, 2019. arXiv:1907.11692.

[13] X. Wu, W. Zheng, X. Xia, D. Lo, Data quality matters: A case study on data label correctness for security bug report prediction, IEEE Transactions on Software Engineering (2022). doi:10.1109/TSE.2021.3063727.

[14] L. Tunstall, N. Reimers, U. E. S. Jo, L. Bates, D. Korat, M. Wasserblat, O. Pereg, Efficient Few-Shot Learning Without Prompts, 2022. doi:10.48550/arXiv.2209.11055.

[15] G. Colavito, F. Lanubile, N. Novielli, Few-shot learning for issue report classification, in: 2023 IEEE/ACM 2nd Int'l Work. on Natural Language-Based Software Eng. (NLBSE), 2023.

[16] X. Hou, Y. Zhao, Y. Liu, Z. Yang, K. Wang, L. Li, X. Luo, D. Lo, J. Grundy, H. Wang, Large language models for software engineering: A systematic literature review, 2023. arXiv:2308.10620.

[17] A. Fan, B. Gokkaya, M. Harman, M. Lyubarskiy, S. Sengupta, S. Yoo, J. M. Zhang, Large language models for software engineering: Survey and open problems, 2023. arXiv:2310.03533.

[18] OpenAI, ChatGPT: Optimizing Language Models for Dialogue, 2022.

[19] G. Colavito, F. Lanubile, N. Novielli, L. Quaranta, Leveraging gpt-like llms to automate issue labeling, in: 2024 IEEE/ACM 21th International Conference on Mining Software Repositories (MSR) (to appear), 2024. doi:10.1145/3643991.3644903.

[20] R. Kallis, M. Izadi, L. Pascarella, O. Chaparro, P. Rani, The nlbse'23 tool competition, in: Proc. of The 2nd Intl. Work. on Natural Language-based Software Engineering (NLBSE'23), 2023.

[21] A. J. Viera, J. M. Garrett, Understanding interobserver agreement: the kappa statistic, Family medicine (2005).

[22] G. Colavito, F. Lanubile, N. Novielli, Few-shot learning for issue report classification, 2023. doi:10.5281/zenodo.7628150.

[23] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei, Language models are few-shot learners, in: Proceedings of the 34th International Conference on Neural Information Processing Systems,

NIPS'20, Curran Associates Inc., Red Hook, NY, USA, 2020.

[24] S. Ouyang, J. M. Zhang, M. Harman, M. Wang, Llm is like a box of chocolates: the non-determinism of chatgpt in code generation, 2023. arXiv:2308.02828.

[25] J. Wei, X. Wang, D. Schuurmans, M. Bosma, b. ichter, F. Xia, E. Chi, Q. V. Le, D. Zhou, Chain-of-thought prompting elicits reasoning in large language models, in: S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, A. Oh (Eds.), Advances in Neural Information Processing Systems, volume 35, Curran Associates, Inc., 2022, pp. 24824–24837.

[26] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, Y. Iwasawa, Large language models are zero-shot reasoners, in: S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, A. Oh (Eds.), Advances in Neural Information Processing Systems, volume 35, Curran Associates, Inc., 2022, pp. 22199–22213.