Classification of Issue Discussions in Open Source Projects Using Deep Language Models *

Şevval Mehder
Boğaziçi University
Turkey
sevval.mehder@boun.edu.tr

Fatma Başak Aydemir
Boğaziçi University
Turkey
basak.aydemir@boun.edu.tr

2022 IEEE 30th International Requirements Engineering Conference Workshops (REW) | 978-1-6654-6000-2/22/$31.00 ©2022 IEEE | DOI: 10.1109/REW56159.2022.00040

***Abstract*—Open source projects are supported by contributors located in different locations and time zones. Issues facilitate asynchronous communication among them, including bug reports, solution suggestions, questions, answers or personal communication. The discussion threads contain rich information about the requirements and the system problems, yet they may be lengthy and lack of structure; making it challenging to identify and link the relevant parts to requirements. Previous work addresses this problem by applying shallow machine learning techniques to automatically classify sentences in a discussion. We propose using the state-of-the-art transformer-based deep language models for this task. The contribution of this paper is three-fold. First, we create a benchmark for the community to ensure standardized inputs for training and testing for this problem. Second, we replicate the state-of-the-art and report the results on this benchmark. Third, we significantly outperform the state-of-the-art with our transformed-based classification approach.**

***Index Terms*—open-source systems, issue classification, requirements tracing, transformer models, deep language models**

I. Introduction

Requirements traceability is a critical task for software development [1] and maintenance for it links requirements with other artifacts such as models, code, and user feedback. Software projects benefits from the efforts towards requirements traceability in multiple ways such as uncovering the coverage of the requirements of a project, reducing the task completion time and improving the quality of software maintenance activities [2]. Such efforts also reduce the number of flaws in software products [3].

Requirements traceability is also essential for open source projects where several contributors located in different locations asynchronously work on the same project. The links among the artifacts such as requirements, commits, code, and issues help the existing and new contributors to keep track of the status of the project without the need of the contributors' synchronising over a face-to-face meeting or a call.

An issue in an open-source project may be a way of asking questions, solving problems, reporting bugs, requesting features, or working on ideas with the other collaborators [4], [5]. As such, the communication over issues is a vital source of information for open source projects [6]. Once an issue is created, the members of the community read and comment on it, creating a discussion under the issue. This discussion may be hard to follow for it may contain member questions, their replies, questions that have already been answered, or off-topic sub-discussions [7]. In a discussion thread the members may greet each other and engage in social conversations, or they may mention other issues and share their opinions on it which complicates finding relevant information within the thread.

Issue-tracking systems (ITSs) list the issues, allow the users label them such as open or closed, and present the discussion under the issue. However, they do not extract rich information available in the discussions. Arya et al. [7] identify 16 types of information in open-source project discussions and build a dataset of the most commented issues and their discussions from three open-source projects. The dataset is annotated with 16 labels identified by the authors. Table I presents these labels.

TABLE I
Types of Information in Issue Discussions Identified by Arya et al. [7]

Information Type	
Solution Discussion	Contribution and Commitment
Usage	Social Conversation
Expected	Bug Reproduction
Motivation	Potential New Issues and Requests
Workarounds	Investigation and Exploration
Future Plan	Task Progress
Action on Issue	Observed Bug Behaviour
Content Management	Testing-Related

Previous work proposes using shallow machine learning and natural language processing (NLP) techniques to automatically classify issues for it is a time-consuming task when conducted manually. Kallis et al. [8] treat the whole issue as a computational unit and classify it using the issue title and the text body. However, an issue may contain different types of information as shown by Arya et al [7]. Instead of treating an issue as a whole, they classify each sentence of an issue and the discussion under it using shallow machine learning techniques.

Transformer based language models are demonstrated to outperform the more traditional techniques for text classification [9]–[11]. This paper seeks to answer the following research question:

2770-6834/22/$31.00 ©2022 IEEE
DOI 10.1109/REW56159.2022.00040

RQ: Can transformer-based language models be used to classify sentences from issue discussions?

To answer our research question we apply BERT [12], DistilBERT [13] and RoBERTa [14] models, report the results, and compare their performance on the dataset shared by Arya et al. [7]. Our results indicate that BERT [12] and RoBERTa [14] transformer-based models significantly outperform the best performing algorithm of Arya et al. [7]. We provide our source code, and the benchmark dataset in a public repository[1]. Our contributions are three-fold.

- C1: To standardize training, validation, and test data we split the dataset and share our splits as a benchmark for future research.
- C2: We replicate the work of Arya et al. [7] and report the performance on the new benchmark.
- C3: We apply and report the performance of transformer-based language models.

The rest of the paper is organized as follows: Section II briefly presents a theoretical background of transformed-based deep language models. Section III describes the dataset, our benchmark, and the replication study. Section IV explain our transformer-based models, and their evaluation. Section V discusses the results, Section VI reviews the main threats to validity. Section VII examines the related work and Section VIII concludes the paper.

II. Background

BERT [12], stands for Bidirectional Encoder Representations from Transformers, is a language model that uses transformer architecture [15] inside. Language models are responsible for estimating text sequences, and BERT [12] has a novel pretraining task called Masked Language Model. Transformers outperform the state-of-the-art solutions for natural language processing tasks and have had a significant impact over the past few years.

Radford et al. [16] propose a concept for Transformer Architecture to fine-tune language models. These models can be used for any downstream NLP tasks after the pre-training process. Several models have been proposed to improve BERT [12] in terms of computational power, training time, and prediction metrics. Several transformer-based models exist. BERT is trained over 16 GB of data from books corpus and Wikipedia, RoBERTa [14] is trained with an additional 144 GB of data comes from the news dataset and web text corpus. DistilBERT [13] uses the same data to train but it learns a distilled version of BERT which approximates it. RoBERTa improves the performance by using more data to train, DistilBERT improves the speed by using a different method called BERT Distillation.

III. Benchmark and Replication

This section introduces the original dataset and our benchmark and reports the results of our replication study.

[1]https://github.com/sevvalmehder/Classification-of-Issue-Discussions-in-Open-Source-Projects

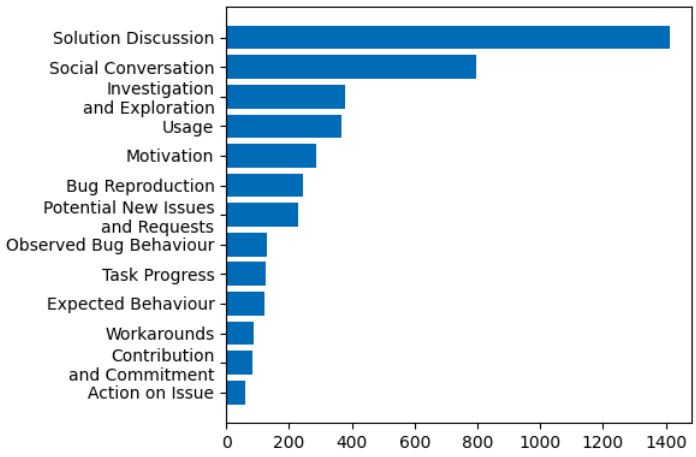

Fig. 1. The label distribution of the dataset

A. Data

The original dataset [7] includes 4656 labeled sentences annotated with 16 information types. It contains sentences from three important AI libraries hosted on Github. The authors selected the five most commented and closed issues, and all sentences are extracted from these issues. According to the analyses of Arya et al. [7] 293 samples from the dataset are duplicates, and three minority classes have only 33 samples in total which is less than 1% of the whole dataset Thus, the authors ignored these 326 samples and 3 classes.

As a result of this process, the benchmark dataset contains 4330 sentences from 13 classes. The issue discussions contain some code blocks, and URLs as external sources. In the original dataset, these parts are replaced with CODE, and URL.

Figure 1 presents the label distribution of the dataset which is quite unbalanced. Three highest frequency labels are Solution Discussion, Social Conversation and Investigation and Exploration. Three lowest frequency labels, which are ignored by the original authors as mentioned above, are Workarounds, Contribution and Commitment, and Action on Issue.

B. Benchmark

In an attempt to mimic the benchmarks used in NLP tasks by the NLP research community which provide the training and test sets separately to standardize the training and testing different approaches, we split the original dataset and share the splits in our replication package. When splitting the dataset we preserve the the label ratios among the annotated data.

The following steps are followed to split the dataset:

1) The X and y vectors are created from the dataset. X represents the text content of the sentences, and the y vector represents the labels of these contents.
2) 'train_test_split' function from sklearn [17] is used to split these X and y vectors into X_train, y_train, X_test and y_test vectors. 'random_state' set as 42 and

'test_size' set as 0.2 that means 20% of the dataset is spared as the test set. The y vector is given as 'stratify' parameter to preserve class distribution.

3) The X and y vectors belonging to the training and test are concatenated. The test and training data are saved as pickle[2] files as the last step.

TABLE II
FEATURES OF TEST AND TRAIN SETS

Feature	Train set	Test set
Length of the longest sentence in terms of character count.	2967	793
Average length of sentences in terms of character count.	87	82
Length of the longest sentence in terms of word count	793	963
Average length of sentences in terms of word count	14	13
Number of unique words	9492	3491
The percentage of sentences including CODE	0.16	0.20
The percentage of sentences including URL	0.028	0.017

Table II compares the training and test set according to features focused on describing the dataset's characteristics. We use the length features in terms of character and word counts. We also used the information of how many elements in the sets contain CODE and URL to explore the dataset's attributes. This comparison shows that the benchmark dataset's test and train parts have similar features.

C. Replication

The original work has 12 experiments using different models, feature sets, class imbalance handling, and evaluation methods. Logistic Regression and Random Forest classifiers are the models of the experiments. The authors handle the class imbalance problem by adjusting class weights and using the SMOTE method [18]. The first feature set that they run experiments with is the textual feature set, the second is conversational features, and finally, they use both textual and conversational features. They create two scenarios that differ in evaluation manners. They use 5-fold cross-validation in the first scenario and Leave-One-Group-Out cross-validation in the second scenario. In the second scenario, they select 14 issue threads to train the classifier and leave out the one issue for testing the model. They iterate this process 15 times.

Since our version of the classification problem focuses on the textual features, we only replicate the best-performing model on the textual feature set. The model that has higher results is the Logistic Regression model with adjusted class weights. Thus, once we create our benchmark, we run the code of the Logistic Regression model provided by Arya et al. [7]. The original pipeline includes the following steps:

1) Cleaning and preprocessing sentences
2) Transforming sentences to vectors with n-gram range
3) Using these vectors as input to the Logistic regression model

One important point is that the original authors use their own custom tokenizer to perform tokenizing, lemmatization, case folding, and punctuation removal operations. They also identify mentions to GitHub users and replace them with a special token named SCREEN_NAME at the tokenization step. However, the benchmark dataset does not contain this kind of special token. So we ignore this custom tokenizer to create a similar setup between the original work and the replicated one. The best performing model is trained using bi-grams and 10 for the inverse regularization strength.

D. Replication Results

Table III compares the results of our replication and the original values presented in [7]. The results we report as original work belongs to Logistic Regression results for the Leave-One-Group-Out cross-validation scenario, just as the authors reported. The original precision, recall, and F1 scores are the average scores of 15 iterations. And the results we reported as replicated work belongs to the Logistic Regression model, which is trained with the hold-out method over the benchmark dataset. We follow the same steps except for the validation method.

The difference between the original and replicated work is because of the validation method of the original work, which is Leave-One-Group-Out. The leave-One-Group-Out strategy does not guarantee the class distributions of the training data will be similar. Because of this situation, we observe either comparable or improved F1 scores for the replication over the original values. For some labels, the improvement is more prominent, such as Solution Usage, Motivation, Potential New Issues and Requests, and Expected Behavior. In contrast, for other labels, the results are comparable such as Solution Discussion, Social Conversation, and Contribution and Commitment.

TABLE III
PERFORMANCE OF THE ORIGINAL STUDY AND OUR REPLICATION

Label	Original precision	Replicated precision	Original recall	Replicated recall	Original F1 score	Replicated F1 score
Solution Discussion	0.59	0.54	0.65	0.65	0.58	0.59
Social Conversation	0.74	0.73	0.69	0.69	0.70	0.71
Investigation and Exploration	0.47	0.43	0.24	0.51	0.31	0.47
Usage	0.57	0.44	0.08	0.46	0.12	0.45
Motivation	0.44	0.35	0.1	0.31	0.13	0.33
Bug Reproduction	0.53	0.48	0.36	0.45	0.42	0.46
Potential New Issues and Requests	0.1	0.29	0.03	0.24	0.03	0.26
Observed Bug Behaviour	0.23	0.21	0.03	0.12	0.04	0.15
Task Progress	0.35	0.38	0.26	0.32	0.29	0.35
Expected Behaviour	0.71	0.48	0.1	0.40	0.15	0.43
Workarounds	0.51	0.40	0.06	0.11	0.09	0.17
Contribution and Commitment	0.51	0.56	0.31	0.29	0.37	0.38
Action on Issue	0.78	0.88	0.49	0.58	0.58	0.70

[2]https://docs.python.org/3/library/pickle.html

IV. Classification with Deep Language Models

A. Method

For our transformer-based approach, we focus on BERT [12], RoBERTa [14] and DistilBERT [13]. We use the pre-trained *bert-base-uncased* model for BERT, *distilbert-base-uncased* model for DistilBERT, and *roberta-base* for RoBERTa from Huggingface [19]. These are pretrained language models that each one has different benefits in terms of accuracy and computational cost.

Another important decision is selecting the optimizer. Optimization algorithms are responsible for controlling the parameters such as learning rate and weight and reducing the losses to provide more accurate results. We use AdamW [20] as our optimizer. We perform the validation by randomly splitting the training dataset into two: 85 of the sentences are used to train the classifier, and the remaining 15% for testing it. We set the batch size to one and trained over 40 epochs.

To handle the imbalanced data, the logistic regression model uses class weights. We also assign weights to classes for the BERT and variants using the PyTorch [21] library. The class weights are inversely proportional to the number of class elements.

A benchmark, as it is used in NLP, provides a perspective to keep track of the progress in the related field. However, it is not a realistic assumption that each class has the same distribution for training and test sets. A different evaluation setting is required to estimate how the model performs when a new issue comes. Thus, we design two evaluation settings: (1) Using the benchmark dataset and (2) Leave-One-Group-Out cross-validation.

a) Evaluation Setting 1: We trained BERT, DistilBERT, and RoBERTa using the hold-out method with the benchmark dataset in this setting. We aim to observe the deep language model's performance over the benchmark dataset.

b) Evaluation Setting 2: We use all data, which is the combined version of the training and test dataset in this setting. We leave out one issue of the combined dataset from the training process and use the excluded issue as a test set. We train 15 RoBERTa models, iterating this process 15 times. This method allows us to examine how the model will perform when a new issue comes.

B. Results

We examine the performance of the deep language models separately.

a) Evaluation Setting 1: Table IV presents the weighted F1 scores belongs to Replicated Logistic Regression, BERT, DistilBERT, and RoBERTa. The weighted F1 score is calculated by accounting for each class F1 score as weighted by the number of class samples. According to the overall F1 scores, BERT variants outperform the Logistic Regression model. Table V shows the precision, recall, and F1 scores for each label by the Replicated Logistic Regression, BERT, DistilBERT, and RoBERTa.

We conduct statistical testing to confirm that these classifiers indeed have meaningfully different performances. We show the data are not normally distributed in Figure 1 and consider this when selecting the statistical test. Since we examine the performances of the group of three classifiers, we apply the Friedman test [22] to determine whether these classifiers differ statistically significantly. After that, we report the Wilcoxon Sign Test [23] to make a pairwise comparison between the classifiers.

TABLE IV
The weighted F1 scores of trained models

	Logistic Regression	BERT	DistilBERT	RoBERTa
Weighted F1 score	0.51	0.54	0.53	0.54

We set our null hypothesis that "There is no model that performs significantly different than others in the group.". The test results in a p-value for the Friedman Test [22] is 0.027 which is smaller than 0.05 that rejects our null hypothesis. We apply pairwise Wilcoxon [23] tests to determine the models that perform significantly differently. Wilcoxon tests suggest that there is a significant difference in the performance of the Logistic Regression Model and all three transformer-based models. The tests do not suggest a significant difference among the transformer-based models themselves.s

b) Evaluation Setting 2: : Table VI summarizes the F1 scores of 15 runs. We use only RoBERTa model for these 15 runs. Since some classes are not well represented when we leave out one of the issues, the classes except Social Conversation have zero F1 scores in at least one of the 15 runs. The results are nearly suitable with the results of the trained RoBERTa model over the benchmark dataset. For both cases, the maximum score belongs to the Social Conversation class. The Potential New Issues and Requests and the Observed Bug Behaviour classes have the minimum scores.

V. Discussion

Transformers have been significantly used for not only NLP tasks but also the Requirements Engineering (RE) tasks. Yet, the performance of transformer-based models reported in this work is not as good as in other RE works using them [10], [11], [24]. Mekala et al. [9] claim that transformer-based models can achieve higher results on the small dataset. They have two experiments with 1000 and 1242 samples which are smaller than our benchmark dataset. However, we deal with a multi-class problem with 13 classes while they have a binary classification problem. Considering that some of the classes we use have less than 100 samples, we interpret these results as the consequence of the insufficient samples for each label.

Figure 2 presents the confusion matrix of the RoBERTa model, which is trained over the benchmark dataset. According to the matrix, the model tends to classify most sentences as a social conversation. Our observation here is that, even if a sentence contains, for example, a solution discussion, the discussion messages may include a phrase that can be considered a social conversation, making it challenging to label some sentences. For instance, the sample sentence "*Would greatly appreciate*" is labeled as a Solution Discussion;

TABLE V
COMPARISON OF BERT [12], TWO VARIANTS [13], [14] AND THE LOGISTIC REGRESSION [7] IN TERMS OF PRECISION, RECALL AND F1 SCORE

Labels	Replicated Logistic Regression			BERT			DistilBERT			RoBERTa		
	pr	rec	F1	pr	rec	F1	pr	rec	F1	pr	rec	F1
Solution Discussion	0.54	0.65	0.59	0.53	0.73	0.61	0.57	0.72	**0.64**	0.54	0.69	0.61
Expected Behaviour:	0.48	0.40	0.43	0.56	0.40	**0.47**	0.24	0.32	0.28	0.53	0.40	0.45
Usage	0.44	0.46	0.45	0.56	0.47	**0.51**	0.39	0.54	0.45	0.46	0.45	0.46
Social Conversation	0.73	0.69	0.71	0.80	0.81	0.80	0.80	0.74	0.77	0.80	0.82	**0.81**
Contribution and Commitment	0.56	0.29	0.38	0.58	0.41	**0.48**	0.67	0.35	0.46	0.60	0.35	0.44
Bug Reproduction	0.48	0.45	0.46	0.50	0.45	0.47	0.42	0.41	0.41	0.45	0.51	**0.48**
Motivation	0.35	0.31	**0.33**	0.42	0.22	0.29	0.44	0.24	0.31	0.41	0.24	0.30
Potential New Issues and Requests	0.29	0.24	0.26	0.38	0.26	**0.31**	0.33	0.28	**0.31**	0.34	0.24	0.28
Investigation and Exploration	0.43	0.51	**0.47**	0.41	0.37	0.39	0.48	0.33	0.39	0.44	0.37	0.40
Workarounds	0.40	0.11	0.17	1.00	0.17	0.29	0.50	0.11	0.18	0.50	0.28	**0.36**
Observed Bug Behaviour	0.21	0.12	0.15	0.33	0.15	0.21	0.33	0.15	0.21	0.29	0.23	**0.26**
Task Progress	0.38	0.32	0.35	0.42	0.32	0.36	0.55	0.48	**0.51**	0.50	0.32	0.39
Action on Issue	0.88	0.58	0.70	0.80	0.67	0.73	0.70	0.58	0.64	0.82	0.75	**0.78**

TABLE VI
PERFORMANCE OF THE
LEAVE-ONE-GROUP-OUT CROSS VALIDATION.

Label	Max F1	Min F1	Avg F1	St. Dev.
Solution Discussion	0.68	0.00	0.47	0.18
Social Conversation	0.88	0.57	0.74	0.08
Investigation and Exploration	0.60	0.00	0.21	0.24
Usage	0.80	0.00	0.11	0.22
Motivation	0.50	0.00	0.10	0.15
Bug Reproduction	0.58	0.00	0.21	0.24
Potential New Issues and Requests	0.22	0.00	0.06	0.08
Observed Bug Behaviour	0.60	0.00	0.14	0.19
Task Progress	0.86	0.00	0.36	0.18
Expected Behaviour	0.13	0.00	0.01	0.04
Workarounds	0.57	0.00	0.07	0.15
Contribution and Commitment	0.67	0.00	0.29	0.23
Action on Issue	1.00	0.00	0.54	0.30

however, the trained RoBERTa model confuses and predicts a Social Conversation.

Figure 3 presents the matrix of common words. According to the confusion and the common words matrices, the model makes a mistake between two classes if they share many common words. According to the confusion matrix, the model confuses the Solution Discussion and Motivation classes, they also share 885 common words. The model never fails differentiating Solution Discussion and Action on Issue and they share only 176 common words. Furthermore, the F1 scores are higher for Action on Issue even though it has the fewest samples. The reason is that the maximum word count that Action on Issue samples share with another class is 176. But also, we suspect the model is over fitting.

In light of this information, we conclude that the model's performance is associated with the number of common words they share. Some minority classes may be grouped together to improve the model's performance if they share many common words as part of the future work. This reduces the imbalance problem, which poses a challenge for the model. Training binary classifiers for each label using over or undersampling may also further improve the results.

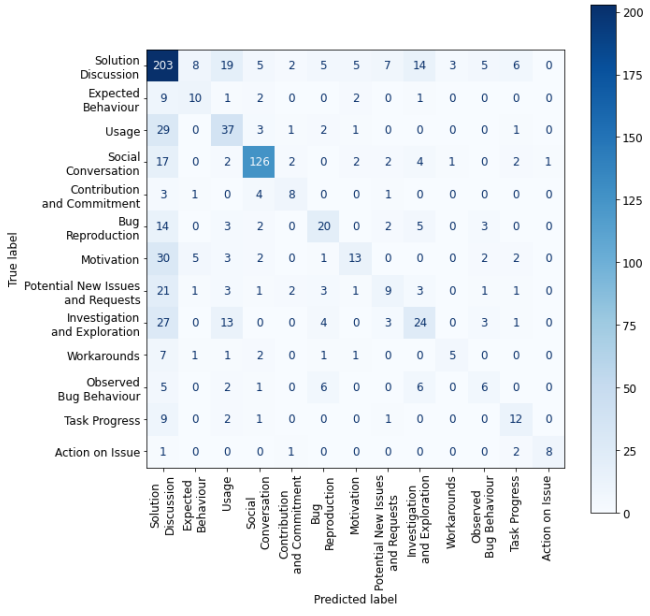

Fig. 2. Confusion matrix of RoBERTa model trained over benchmark dataset

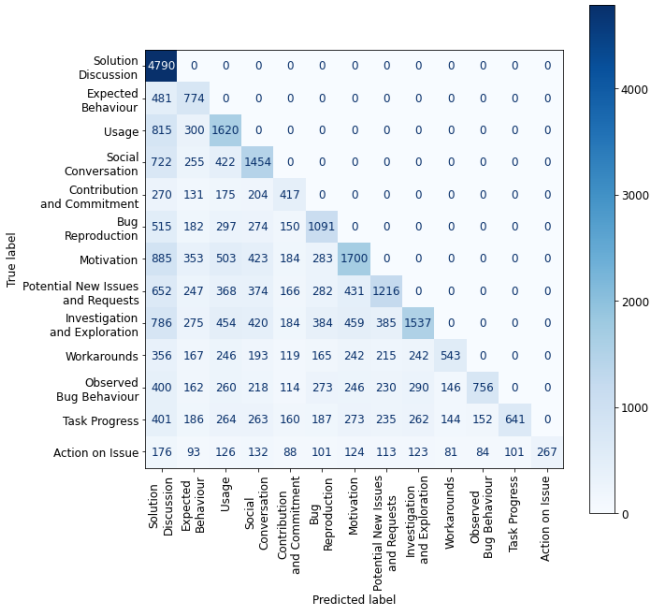

Fig. 3. The number of common words matrix

VI. Threats to Validity

External validity. External validity concerns the generality of the results. The dataset we use is collected from three open-source projects on artificial intelligence. As such we do not claim generality and call the community to study transformer-based models on other type of projects and domains.

Internal validity. We use a readily annotated dataset; therefore we do not reflect our bias to the annotation if there exists any bias. When splitting the data, we make use of the publicly available tools, and publicly share our code to prove the authors do not intervene with the splits for training and testing.

Construct validity. Our replication is not an exact replication for we rely on the pre-trained tokenizers of the transformer-based models and the original study uses its own custom tokenizer that also preprocesses the data. This change in the tokenization step may affect the replication results of logistic regression.

Conclusion validity. To mitigate any threat to validity of our interpretation of the results we perform statistical tests. First, the Friedman Test indicates that there is a significantly performing model among the models. Second the pairwise Wilcoxon tests show that BERT [12] and RoBERTa [14] transformer-based models significantly outperform the Logistic Regression Model.

VII. Related Work

This work is closely related to previous studies concentrating on requirement traceability and the classification of requirement-related texts.

Researchers use the issues classification method for different objectives. The issues are used to determine software feature requests [5]. They classify issues into three classes: requests, clarifications, and solution proposals. Their technique can reasonably detect whether the issue contains any software feature request; however, their model cannot identify the exact sentence that proposes a feature request. Kallis et al. [8] build a tool to classify Github issues as bugs, enhancements, and questions using fastText [25]. They train their model using both the issue title and the body and evaluate the tool for over 30,000 GitHub issues. Again, this study can assign a class to the whole issue and not perform classification at the sentence level.

Most issue classification methods classify the whole issue instead of classifying sentences in the issues [8], [26], [27]. They assign a label for the first title and body that starts the issue discussion for an open issue. On the other hand, we focused on extracting important information types from the sentences in issue discussions.

The other related topic is the classification of requirement-related texts. The literature has established that deep learning-based approaches have significant performance in RE, just as natural language processing in recent years. Dalpiaz et al. [28] focused on the requirement documents to classify and work on distinguishing between functional and non-functional (quality) requirements from the PROMISE dataset. They build a machine learning classifier using interpretable features such as dependency types instead of the features that do not give any semantical clue like n-grams or POS n-grams which are popular for classification problems. After that Hey et al. [10] proposed NoRBERT using the same dataset, PROMISE. Their method of fine-tuning BERT to apply deep learning methods for the poor generalizability problem outperforms recent approaches using machine learning solutions.

In order to compare ML techniques with DL-based classification techniques made by Sainani et al. [11]. They compare Support Vector Machines, Random Forest, Naïve Bayes, BiLSTM(Bidirectional Long Short-Term Memory), and BERT to determine the types of the requirements from software contracts. They trained the models over 20 contracts and used multilabel classification techniques for their classification. As a result, they prove that BERT produced better results for each requirement type even if they have a lesser number of samples in the dataset.

Furthermore Fischbach et al. [24] made an experiment setup for automatic causality detection and compared different models from rule-based methods to deep learning-based methods. Their pre-trained BERT [12] model with the dependency types of the tokens, which they called BERT_{DEP} gives a higher F1 score and outperforms the Random Forest. Another recent work that proves the success of the deep learning-based models belongs to Mekala et al. [9]. They use the fine-tuning BERT method to classify requirements in user feedback to determine whether a part of the feedback is relevant from an RE perspective with a small dataset. This study also proves that BERT is reliable in determining the relevant portions of user feedback.

VIII. Conclusions

This paper aims at applying state-of-the-art deep language models to the issue discussion classification problem for open-source projects. First, we create a benchmark from a state-of-the-art dataset, replicate the approach of Arya et al. [7], and train three BERT variants for this task. Our results indicate that BERT variants outperform the shallow machine learning method employed by the original study.

Future work includes balancing the dataset with more samples for the minority classes. Collecting and classifying issue classifications from open-source projects in other domains than artificial intelligence helps us test the generality of the methods.

References

[1] T. Gorschek, P. Garre, S. Larsson, and C. Wohlin, "A model for technology transfer in practice," *IEEE software*, vol. 23, no. 6, pp. 88–95, 2006.

[2] P. Mäder and A. Egyed, "Do developers benefit from requirements traceability when evolving and maintaining a software system?" *Empirical Software Engineering*, vol. 20, no. 2, pp. 413–441, 2015.

[3] P. Rempel and P. Mäder, "Preventing defects: The impact of requirements traceability completeness on software quality," *IEEE Transactions on Software Engineering*, vol. 43, no. 8, pp. 777–797, 2016.

[4] M. Ohira, Y. Kashiwa, Y. Yamatani, H. Yoshiyuki, Y. Maeda, N. Limsettho, K. Fujino, H. Hata, A. Ihara, and K. Matsumoto, "A dataset of high impact bugs: Manually-classified issue reports," in *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*. IEEE, 2015, pp. 518–521.

[5] T. Merten, M. Falis, P. Hübner, T. Quirchmayr, S. Bürsner, and B. Paech, "Software feature request detection in issue tracking systems," in *2016 IEEE 24th International Requirements Engineering Conference (RE)*. IEEE, 2016, pp. 166–175.

[6] T. Merten, B. Mager, P. Hübner, T. Quirchmayr, B. Paech, and S. Bürsner, "Requirements communication in issue tracking systems in four open-source projects." in *REFSQ workshops*, 2015, pp. 114–125.

[7] D. Arya, W. Wang, J. L. Guo, and J. Cheng, "Analysis and detection of information types of open source software issue discussions," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 2019, pp. 454–464.

[8] R. Kallis, A. Di Sorbo, G. Canfora, and S. Panichella, "Predicting issue types on github," *Science of Computer Programming*, vol. 205, p. 102598, 2021.

[9] R. R. Mekala, A. Irfan, E. C. Groen, A. Porter, and M. Lindvall, "Classifying user requirements from online feedback in small dataset environments using deep learning," in *2021 IEEE 29th International Requirements Engineering Conference (RE)*. IEEE, 2021, pp. 139–149.

[10] T. Hey, J. Keim, A. Koziolek, and W. F. Tichy, "Norbert: Transfer learning for requirements classification," in *2020 IEEE 28th International Requirements Engineering Conference (RE)*. IEEE, 2020, pp. 169–179.

[11] A. Sainani, P. R. Anish, V. Joshi, and S. Ghaisas, "Extracting and classifying requirements from software engineering contracts," in *2020 IEEE 28th International Requirements Engineering Conference (RE)*. IEEE, 2020, pp. 147–157.

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[13] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.

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

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[16] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[18] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[19] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz *et al.*, "Huggingface's transformers: State-of-the-art natural language processing," *arXiv preprint arXiv:1910.03771*, 2019.

[20] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[21] R. Collobert, K. Kavukcuoglu, and C. Farabet, "Torch7: A matlab-like environment for machine learning," in *BigLearn, NIPS Workshop*, 2011.

[22] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *Journal of the american statistical association*, vol. 32, no. 200, pp. 675–701, 1937.

[23] F. Wilcoxon, "Individual comparisons by ranking methods," in *Breakthroughs in statistics*. Springer, 1992, pp. 196–202.

[24] J. Fischbach, J. Frattini, A. Spaans, M. Kummeth, A. Vogelsang, D. Mendez, and M. Unterkalmsteiner, "Automatic detection of causality in requirement artifacts: the cira approach," in *International Working Conference on Requirements Engineering: Foundation for Software Quality*. Springer, 2021, pp. 19–36.

[25] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," *arXiv preprint arXiv:1607.01759*, 2016.

[26] M. L. Siddiq and J. C. Santos, "Bert-based github issue report classification," 2022.

[27] A. Nadeem, M. U. Sarwar, and M. Z. Malik, "Automatic issue classifier: A transfer learning framework for classifying issue reports," in *2021 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. IEEE, 2021, pp. 421–426.

[28] F. Dalpiaz, D. Dell'Anna, F. B. Aydemir, and S. Çevikol, "Requirements classification with interpretable machine learning and dependency parsing," in *2019 IEEE 27th International Requirements Engineering Conference (RE)*. IEEE, 2019, pp. 142–152.