Few-Shot Issue Report Classification with Adapters

Fahad Ebrahim
The University of Warwick
Coventry, UK
Fahad.Ebrahim@warwick.ac.uk

Mike Joy
The University of Warwick
Coventry, UK
M.S.Joy@warwick.ac.uk

ABSTRACT

The automation of the classification of issue reports helps to improve the efficiency of the software tracking cycle. This task can be considered a multi-class classification problem. Therefore, this work is a participation in the NLBSE'24 Issue Report Classification competition.

The paper introduces a lightweight model called AdaptIRC that uses adapters. These adapters add additional trainable layers instead of tuning the whole pre-trained model. Then, they are attached to the transformer to predict classes of the unseen testing data. The dataset provided contained 3,000 reports divided equally for training and testing, having five different repositories and three classes. The newly developed model has achieved an overall F1 score of 0.8934 that exceeded the baseline of an F1 score of 0.827, an approximate increase of 8%.

CCS CONCEPTS

• **Software and its engineering → Software maintenance tools**; **Software configuration management and version control systems**; **Maintaining software**; **Software version control**.

KEYWORDS

Issue Report Classification, Multi-Class Classification, Adapters, Pre-Trained Models, Few-Shot Learning, Software Engineering, Parameter Efficient Fine Tuning.

ACM Reference Format:
Fahad Ebrahim and Mike Joy. 2024. Few-Shot Issue Report Classification with Adapters. In *2024 ACM/IEEE International Workshop on NL-based Software Engineering (NLBSE '24), April 20, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3643787.3648039

1 INTRODUCTION

Tracking software issues is one step in the software maintenance cycle. It includes manually classifying software report issues, which is time-consuming and demanding. Therefore, automating this task should result in increased efficiency.

This work participates in "The NLBSE'24 Tool Competition" [5], creating an issue report classification tool. The provided dataset includes 3,000 reports, and the classes are either bug, enhancement, or question taken from five different repositories. In addition, the authors provide a baseline [2] using few-shot learning with an overall weighted score of 0.827. Previous work published in the report classification competition has involved [6] and [7] which have used FastText [4] for classification in the created Ticket Tagger system.

Fine-tuning a pre-trained model for a certain task requires adjusting the weights of the entire structure. This requires resources and time. Thus, using adapters with the pre-trained models reduces the computations and produces competitive results. Adapters [12] add additional layers, and the weights of these layers must be adjusted. So, instead of tuning the whole model, only the adapter layers would be trained. Adapters are also considered to be few-shot learners [1].

This work applies adapters on top of a RoBERTa [8] transformer after a few preprocessing steps for the task of issue report classification creating a model called AdaptIRC with an improved average score of 0.8934.

The paper is organised as follows: section 2 covers the tool creation, section 3 presents the results, and section 4 concludes.

2 TOOL CREATION

A preliminary step before creating the tool is to analyse the dataset. It can be pre-processed by removing the parts that would not be useful during training or testing. The new model (AdaptIRC) starts with training an adapter module along with a classification head, and then attaches them to a RoBERTa transformer model to predict the unseen test cases. A simple abstract architecture of the developed models can be seen in Figure 1. Each step is illustrated in the following subsections.

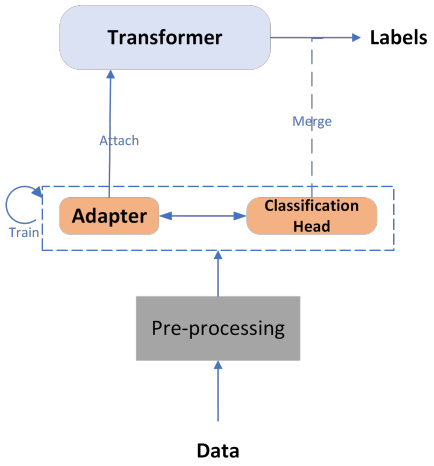

Figure 1: A high-level architecture of the developed model (AdaptIRC). The adapter and classification heads are the trainable entities.

NLBSE '24, April 20, 2024, Lisbon, Portugal

ACM ISBN 979-8-4007-0576-2/24/04
https://doi.org/10.1145/3643787.3648039

Table 1: A sample issue report

repo	opencv/opencv
created_at	2022-01-15 02:39:22
label	feature
title	Reading BigTiff images
body	**Merge with extra: https://github.com/opencv/opencv_extra/pull/952** resolves #18717 ### Pull Request Readiness Checklist See details at https://github.com/opencv/opencv/wiki/How_to_contribute#making-a-good-pull-request - [ ] I agree to contribute to the project under Apache 2 License. - [ ] To the best of my knowledge, the proposed patch is not based on a code under GPL or other license that is incompatible with OpenCV - [ ] The PR is proposed to proper branch - [ ] There is reference to original bug report and related work - [ ] There is accuracy test, performance test and test data in opencv_extra repository, if applicable Patch to opencv_extra has the same branch name. - [ ] The feature is well documented and sample code can be built with the project CMake

2.1 Dataset

The dataset consists of 3,000 issue reports and an example issue report can be seen in Table 1. The columns in the dataset are the following.

- Repository: the dataset includes 5 repositories: facebook/react, tensorflow/tensorflow, microsoft/vscode, bitcoin/bitcoin, and opencv/opencv.
- Title: the title of the issue report.
- Creation date: the date on which the issue was created.
- Body: the content of the issue report.
- Label: the class of an issue, being a bug, an enhancement, or a question.

So, 3,000 reports with 5 categories produced 600 reports per repository for both training and testing, divided equally. Therefore, there were 300 training reports per category and 300 instances for testing. As a result, few-shot learning algorithms are more suitable for this task because of limited training data.

2.2 Pre-Processing

The dataset includes lengthy texts and unnecessary words. Therefore, the following pre-processing steps with regular expressions were utilised.

- Remove new lines, multiple tabs, and multiple spaces.
- Remove links and errors.
- Remove special characters except for question marks.
- Remove the text written in triple quotes.

Both the title and text were pre-processed, and then they were concatenated to be the input text. This text was tokenised and fed into the transformer model.

2.3 Adapters

The general single encoder of the transformer architecture [15] can be seen on the left of Figure 2. It starts by converting the input into embeddings and adding the positional encoding to them. Then, the new resulting input is fed into a multi-head attention and an (add and norm) block. Then, the output of the (add and norm) layer is fed to the feed-forward network and another (add and norm) layer. Adapters aim to add additional layers to the transformer models, and these layers would be tuned. There are various adapter methods. One of the first architectures that created adapters for natural language processing [3] added two additional layers before and after the feed-forward layer in the transformer blocks. Another technique [9, 11] adds a single adapter layer as in the right of Figure 2 and has minor differences in accuracy. Therefore, the latter adapter was used in this work, as it has fewer weights to adjust.

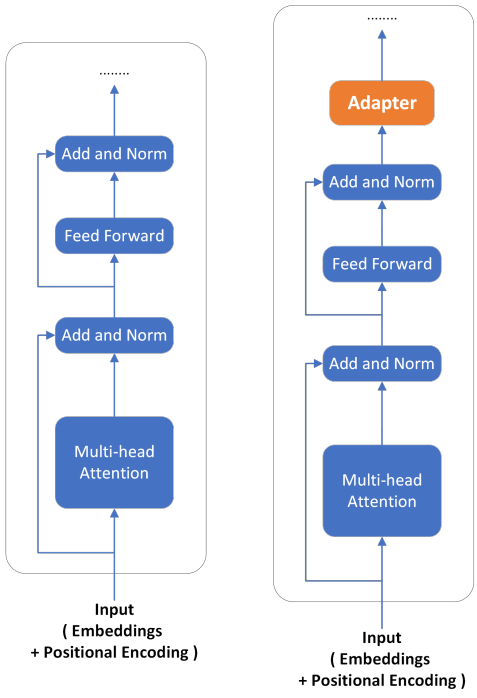

Figure 2: The transformer encoder architecture. The left represents the general architecture [15]. The right represents the adapter type used in this work [9, 11].

Table 2: Results

Repository/Model	Baseline				AdaptIRC		
	Label	Precision	Recall	F1-Score	Precision	Recall	F1-Score
facebook/react	Bug	0.9048	0.9500	0.9268	**0.9159**	**0.98**	**0.9468**
	Feature	0.8491	0.9000	0.8738	**0.9126**	**0.94**	**0.9261**
	Question	0.8652	0.7700	0.8148	**0.9556**	**0.86**	**0.905**
	Average	0.8729	0.8733	0.8718	**0.928**	**0.9267**	**0.926**
tensorflow/tensorflow	Bug	0.9565	**0.8800**	**0.9167**	**0.9205**	0.81	0.8617
	Feature	**0.8558**	**0.8900**	**0.8725**	0.8173	0.85	0.833
	Question	0.7885	0.8200	0.8039	**0.7963**	**0.86**	**0.8269**
	Average	**0.8669**	**0.8633**	**0.8644**	0.8447	0.84	0.8407
microsoft/vscode	Bug	0.8485	0.8400	0.8442	**0.891**	**0.9**	**0.8955**
	Feature	0.7627	0.9000	0.8257	**0.901**	**0.91**	**0.9055**
	Question	0.8916	0.7400	0.8087	**0.9592**	**0.94**	**0.9495**
	Average	0.8343	0.8267	0.8262	**0.9171**	**0.9167**	**0.9168**
bitcoin/bitcoin	Bug	0.7604	0.7300	0.7449	**0.8958**	**0.86**	**0.87755**
	Feature	0.8723	0.8200	0.8454	**0.9293**	**0.92**	**0.9246**
	Question	0.6455	0.7100	0.6762	**0.8476**	**0.89**	**0.8683**
	Average	0.7594	0.7533	0.7555	**0.8909**	**0.89**	**0.8902**
opencv/opencv	Bug	0.7619	0.8000	0.7805	**0.8654**	**0.9**	**0.88235**
	Feature	0.8842	0.8400	0.8615	**0.927**	**0.89**	**0.9082**
	Question	0.8100	0.8100	0.8100	**0.89**	**0.89**	**0.89**
	Average	0.8187	0.8167	0.8173	**0.8942**	**0.8933**	**0.8935**
Overall	Bug	0.8464	0.8400	0.8426	**0.8977**	**0.89**	**0.8928**
	Feature	0.8448	0.8700	0.8558	**0.8975**	**0.902**	**0.8995**
	Question	0.8001	0.7700	0.7827	**0.8897**	**0.888**	**0.888**
	Average	0.8305	0.8267	0.8270	**0.895**	**0.8933**	**0.8934**

2.4 Training

For a better evaluation of the generalisation of the developed model, the training data sets were divided into training and validation sets. Since the training sets were small, the percentage of validation sets was selected to be 30% of the training sets. This led to two subsets: training with 210 reports per repository and validation with 90 instances per repository.

The following are the steps for training the AdaptIRC model.

(1) The adapter module is initialised with random weights.
(2) The selected adapter module consists of two feed-forward layers that are small neural networks.
(3) The weights of these two layers are fine-tuned, whereas the other transformer model weights are frozen.
(4) While the adapter module is trained, a Classification Head (CH) is trained simultaneously.
(5) The CH consists of a combination of linear and dropout layers that are part of a feed-forward network.
(6) The adapter is attached to the model.
(7) The CH is added to the transformer's output and maps the generated mean-pooling embeddings to a single class, which is a bug, enhancement, or question.
(8) A single adapter is trained for each repository, leading to five different adapters.

For adapter training, the parameters were the following: the learning rate was 1×10^{-4}, batch size was 32, and the number of epochs was 200. The metric for choosing the best model was the validation loss. The maximum length of the input was set to 256.

The T4 Nvidia Graphical Processing Unit (GPU) with a Random Access Memory (RAM) of 12GB was used for training and testing. The full code runs within 1 to 2 hours. Each adapter takes around 15 minutes to be trained.

One feature of adapters is their modularity and composability [10]. Another feature is that they are lightweight as there are fewer parameters to train. For example, the RoBERTa model is around 500MB, while the adapters are a few megabytes around 6MB. This makes it very convenient to store and is easily accessible. For instance, in this work, we have five repositories, so training the full models would yield five models with each having a size of around 500MB, while in adapters, we have a single model of size 500MB and five adapters with each of size around 6MB. This requires less storage and computation.

3 RESULTS

The evaluation of the model in the test set was based on precision, recall, and F1-score as presented in equations 1, 2, and 3, respectively. For the ranking, it was based on the overall average F1 score.

$$precision = \frac{TP}{TP + FP} \tag{1}$$

$$recall = \frac{TP}{TP + FN} \tag{2}$$

$$f1score = 2 * \frac{precision * recall}{precision + recall} \quad (3)$$

The baseline along with the results of our model can be seen in Table 2 which shows the precision, recall, and f1 score for each class in each repository both in the baseline and in AdaptIRC. The baseline [2] utilised SetFit few-shot learning [14] and Sentence Transformers [13] for their work. The overall F1 scores in AdaptIRC facebook/react, tensorflow/tensorflow, microsoft/vscode, bitcoin/bitcoin, and opencv/opencv were 0.926, 0.84, 0.917, 0.89 and 0.89, respectively. Therefore, the newly developed model exceeds the baseline in all metrics except for the tensorflow/tensorflow repository. A potential reason could be that the tensorflow/tensorflow repository had longer texts than the other repositories even with pre-processing. Furthermore, for the overall average F1 score, AdaptIRC with a score of 0.893 exceeds the baseline score with a score of 0.827 by approximately 8%. This indicates that the adapters were as robust as few-shot learners in the task of issue report classification. One limitation of this work was the limited maximum token size. It was selected to be 256 to be faster in training, it would not consume high resources, and this work would not yield high accuracy if the important parts of the text were at the end of longer sequences of input.

4 CONCLUSION

In this paper, we participated in the Issue Report Classification Competition, creating a lightweight model called AdaptIRC that utilised adapters trained and activated with the RoBERTa transformer with proper preprocessing. The newly developed model achieved an F1 score of 0.89 that outperforms the baseline with an F1 score of 0.827. The code and the created adapters are publicly shared [1].

For future work, different configurations can be explored. Evaluating and comparing experiments with several adapter methods and several pre-trained models could be very productive future work. In addition, it would be interesting to compare the results of full fine-tuning and different adapters. A limitation of this work was that training an adapter per repository is not ideal. Adapters can also be used for transfer learning, so, another possible direction is to explore the performance of a single adapter on different repositories. Another experiment would be to use the full data for training instead of splitting the data into training and evaluation sets, and to monitor the performance over the testing set.

5 ACKNOWLEDGEMENT

The authors sincerely thank the reviewers for their insightful comments and helpful suggestions, which greatly improved the clarity and quality of the article.

REFERENCES

[1] Tilman Beck, Bela Bohlender, Christina Viehmann, Vincent Hane, Yanik Adamson, Jaber Khuri, Jonas Brossmann, Jonas Pfeiffer, and Iryna Gurevych. 2022. AdapterHub Playground: Simple and Flexible Few-Shot Learning with Adapters. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, Valerio Basile, Zornitsa Kozareva, and Sanja Stajner (Eds.). Association for Computational Linguistics, Dublin, Ireland, 61–75. https://doi.org/10.18653/v1/2022.acl-demo.6

[2] Giuseppe Colavito, Filippo Lanubile, and Nicole Novielli. 2023. Few-Shot Learning for Issue Report Classification. In *2023 IEEE/ACM 2nd International Workshop on Natural Language-Based Software Engineering (NLBSE)*. 16–19. https://doi.org/10.1109/NLBSE59153.2023.00011

[3] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning*. PMLR, 2790–2799.

[4] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. 2016. Fasttext. zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651* (2016).

[5] Rafael Kallis, Giuseppe Colavito, Ali Al-Kaswan, Luca Pascarella, Oscar Chaparro, and Pooja Rani. 2024. The NLBSE'24 Tool Competition. In *Proceedings of The 3rd International Workshop on Natural Language-based Software Engineering (NLBSE'24)*.

[6] Rafael Kallis, Andrea Di Sorbo, Gerardo Canfora, and Sebastiano Panichella. 2019. Ticket Tagger: Machine Learning Driven Issue Classification. In *2019 IEEE International Conference on Software Maintenance and Evolution, ICSME 2019, Cleveland, OH, USA, September 29 - October 4, 2019*. IEEE, 406–409. https://doi.org/10.1109/ICSME.2019.00070

[7] Rafael Kallis, Andrea Di Sorbo, Gerardo Canfora, and Sebastiano Panichella. 2021. Predicting issue types on GitHub. *Science of Computer Programming* 205 (2021), 102598. https://doi.org/10.1016/j.scico.2020.102598

[8] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).

[9] Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2021. AdapterFusion: Non-Destructive Task Composition for Transfer Learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, Paola Merlo, Jorg Tiedemann, and Reut Tsarfaty (Eds.). Association for Computational Linguistics, Online, 487–503. https://doi.org/10.18653/v1/2021.eacl-main.39

[10] Jonas Pfeiffer, Sebastian Ruder, Ivan Vulić, and Edoardo Maria Ponti. 2023. Modular deep learning. *arXiv preprint arXiv:2302.11529* (2023).

[11] Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020. MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (Eds.). Association for Computational Linguistics, Online, 7654–7673. https://doi.org/10.18653/v1/2020.emnlp-main.617

[12] Clifton Poth, Hannah Sterz, Indraneil Paul, Sukannya Purkayastha, Leon Engländer, Timo Imhof, Ivan Vulić, Sebastian Ruder, Iryna Gurevych, and Jonas Pfeiffer. 2023. Adapters: A Unified Library for Parameter-Efficient and Modular Transfer Learning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Yansong Feng and Els Lefever (Eds.). Association for Computational Linguistics, Singapore, 149–160. https://doi.org/10.18653/v1/2023.emnlp-demo.13

[13] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (Eds.). Association for Computational Linguistics, Hong Kong, China, 3982–3992. https://doi.org/10.18653/v1/D19-1410

[14] Lewis Tunstall, Nils Reimers, Unso Eun Seo Jo, Luke Bates, Daniel Korat, Moshe Wasserblat, and Oren Pereg. 2022. Efficient few-shot learning without prompts. *arXiv preprint arXiv:2209.11055* (2022).

[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[1] https://github.com/FahadEbrahim/AdaptIRC