Generating Software Tests for Mobile Applications Using Fine-Tuned Large Language Models

Jacob Hoffmann
AIFB - BIS
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
Jacob.hoffmann@partner.kit.edu

Demian Frister
AIFB - BIS
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
Demian.frister@kit.edu

ABSTRACT

Motivation. Software tests are a necessity in the development of software to secure functionality, reliability, and usability [10]; however, these tests are costly and time-consuming [6]. Although tool support for software testing has advanced, there remains considerable potential for enhancement. Many software tests are still devised manually, with the creation of unit tests being particularly laborious. Automating the generation of test cases is promising for streamlining this aspect of software testing [6].

Large Language Models (LLMs) have exhibited capabilities in code generation [11, 13–15], test case generation [17], and various other domains [11]. The advancement of model performance of transformer-based LLMs is mainly achieved by expanding the model size in line with an increase in training data size [7, 8]. However, this approach leads to high computational costs which can only be afforded by corporations with significant financial resources. This highlights the need for transformer-based LLMs that perform well on a specific downstream task and are also cost-efficient. Addressing this, we focused on supervised fine-tuning (SFT) of more resource-efficient transformer-based LLMs LLaMA 2 13B, Code Llama 13B, and Mistral 7B for the specific downstream task of generating test cases for mobile applications.

Research questions. This work investigated: Does SFT enhance the capabilities of a transformer-based LLM in the specific downstream task of generating test cases for mobile applications while being cost-efficient and runnable on standard consumer hardware? Does the fine-tuned model outperform other state-of-the-art models in the task of test generation for mobile applications?

Approach. Our approach is a modification of the ATHENATEST approach [16]. However, our approach focuses on supervised fine-tuning (SFT) on both pre-trained and already fine-tuned transformer-based LLMs for the task of test case generation for mobile applications in Dart.

The approach involves three steps, as illustrated in Figure 1. Firstly, a labeled dataset of corresponding input-output pairs (X, Y) was obtained to model the conditional probability $P(Y|X;\theta)$ [9, 12]. Dart code and corresponding test files were extracted from open-source GitHub repositories using Google BigQuery. These files were then matched using regular expressions, ensuring that each code file was matched with its corresponding test file based on matching base filenames. The dataset underwent quality filtering and deduplication, resulting in 16,252 input-output pairs, which was then divided into training (90%) and validation (10%) sets. The training set of the dataset consists of a total of 88.5M tokens using the LLaMA tokenizer.

Secondly, for SFT on the downstream task of test generation, models were selected based on their code generation capabilities, as indicated by the pass@1 score on the HumanEval [2] and MBPP [1] benchmark, their parameter sizes, and the extent to which they had been trained on Dart data. In model selection, open-source models capable of running on cost-efficient consumer hardware with code generation abilities were primarily chosen.

Thirdly, in the SFT process, the test generation task was represented as translation task, in line with ATHENATEST [16]. This is achieved by employing the following structured prompt format for SFT [9]:

"{prefix_prompt} ### Code: {code} ### Test: {test}"

In this work, there was no prefix prompt used during SFT.

Fine-tuning. The fine-tuning was conducted on a single GPU system using Flash Attention 2 [3] and the QLoRA method [4] to reduce memory size and the number of trainable parameters. The fine-tuning process varied in duration up to 32 hours, resulting in total emissions of 13.099 kgCO2eq [5].

Experimental Results. The performance of *TestGen-Dart* models was evaluated for their unit testing capabilities in Dart, in comparison to base models LLaMA 2 13B, Code Llama 13B, and Mistral 7B. The models were loaded in both float16 and 4-bit quantization configurations, and the evaluation involved nine different Dart files, encompassing 42 test cases. The results were obtained in a zero-shot setting using a structured prompt format, as described in the approach section. This included a prefix prompt instructing the models to generate unit tests: *"Generate unit tests in Dart for the following class. The unit test should be structured with the 'test' function, an appropriate description, and an assertion 'expect' within the function to validate the test case."* The generated unit tests were classified into three categories: syntax errors (SE), syntactic correctness (SC), and functional correctness (FC). In a 4-bit quantization configuration, *TestGen-Dart_v0.2* enhanced the generation of syntactically correct unit tests by

AST '24, April 15–16, 2024, Lisbon, Portugal

ACM ISBN 979-8-4007-0588-5/24/04.
https://doi.org/10.1145/3644032.3644454

15.38% and functionally correct unit tests by 16.67%, compared to the underlying base model, Code Llama 13B. Additionally, *TestGen-Dart_v0.2* demonstrated superior performance in the 16-bit configuration. This evidenced that supervised fine-tuning (SFT) increases the capability of transformer-based LLMs in a specific downstream task, in this instance, generating test cases for mobile applications, addressing the first research question posed in this work. Additionally, *TestGen-Dart_v0.2* outperformed the other state-of-the-art models of interest LLaMA 2 13B and Mistral 7B in that task, addressing the second research question.

Conclusion. This work demonstrates that SFT enhances the capability of transformer-based LLMs in generating test cases for mobile applications in Dart. Furthermore, the 13B parameter size of the *TestGen-Dart* enables it to run locally on standard consumer hardware, potentially making it a cost-efficient and privacy-friendly testing assistant for software developers by avoiding an external server connection to run the model.

Outlook. Future work currently in progress may expand this approach to other programming languages and refine *TestGen-Dart's* performance by using higher-quality fine-tuning data either synthetic or human-annotated. Additionally, the evaluation method may be enhanced by using *TestGen-Dart* for generating test cases for dummy applications and measuring code coverage.

KEYWORDS

Software Testing, Mobile Testing, Machine Learning, Large Language Models

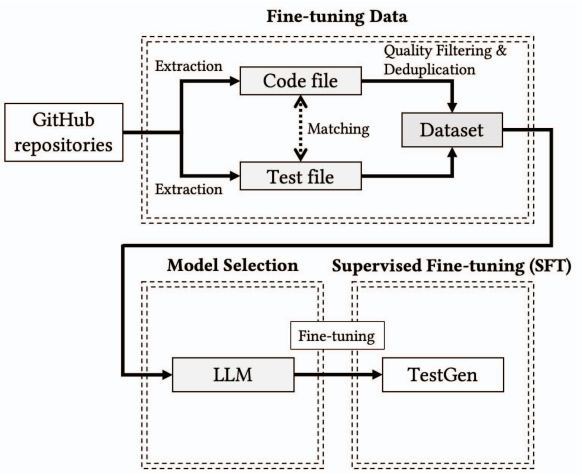

Figure 1: Three steps of the TestGen approach. Modification of the ATHENATEST approach [16].

ACKNOWLEDGMENTS

This work was supported by the Helmholtz Association's Initiative and Networking Fund on the HAICORE@FZJ partition. The dataset for the evaluation was obtained by Khalil Sakly.

REFERENCES

[1] Austin, J., Odena, A., Nye, M., Bosma, M., Michalewski, H., Dohan, D., Jiang, E., Cai, C., Terry, M., Le, Q. and Sutton, C. 2021. Program Synthesis with Large Language Models. arXiv:2108.07732. Retrieved from http://arxiv.org/abs/2108.07732.

[2] Chen, M. et al. 2021. Evaluating Large Language Models Trained on Code. Retrieved from http://arxiv.org/abs/2107.03374.

[3] Dao, T., Fu, D.Y., Ermon, S., Rudra, A. and Ré, C. 2022. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. Retrieved from http://arxiv.org/abs/2205.14135.

[4] Dettmers, T., Pagnoni, A., Holtzman, A. and Zettlemoyer, L. 2023. QLoRA: Efficient Finetuning of Quantized LLMs. arXiv: 2305.14314. Retrieved from http://arxiv.org/abs/2305.14314.

[5] Ember 2022. Electricity Data Explorer. Retrieved from https://ember-climate.org/data/data-tools/data-explorer/.

[6] Gamido, H. and Gamido, M. 2019. Comparative Review of the Features of Automated Software Testing Tools. *International Journal of Electrical and Computer Engineering*. 9, (Oct. 2019), 4473–4478. DOI:https://doi.org/10.11591/ijece.v9i5.pp4473-4478.

[7] Jiang, A.Q. et al. 2023. Mistral 7B. arXiv: 2310.06825. Retrieved from http://arxiv.org/abs/2310.06825.

[8] Kaplan, J., McCandlish, S., Henighan, T., Brown, T.B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J. and Amodei, D. 2020. Scaling Laws for Neural Language Models. arXiv:2001.08361. Retrieved from http://arxiv.org/abs/2001.08361.

[9] Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H. and Neubig, G. 2021. Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. arXiv:2107.13586. Retrieved from http://arxiv.org/abs/2107.13586.

[10] Nidhra, S. 2012. Black Box and White Box Testing Techniques - A Literature Review. *International Journal of Embedded Systems and Applications*. 2, 2 (Jun. 2012), 29–50. DOI:https://doi.org/10.5121/ijesa.2012.2204.

[11] OpenAI 2023. GPT-4 Technical Report. arXiv: 2303.08774. Retrieved from http://arxiv.org/abs/2303.08774.

[12] Radford, A., Narasimhan, K., Salimans, T. and Sutskever, I. 2018. Improving Language Understanding by Generative Pre-Training.

[13] Rozière, B. et al. 2023. Code Llama: Open Foundation Models for Code. arXiv: 2308.12950. Retrieved from http://arxiv.org/abs/2308.12950.

[14] Touvron, H. et al. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. arXiv:2307.09288. Retrieved from http://arxiv.org/abs/2307.09288.

[15] Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E. and Lample, G. 2023. LLaMA: Open and Efficient Foundation Language Models. arXiv:2302.13971. Retrieved from http://arxiv.org/abs/2302.13971.

[16] Tufano, M., Drain, D., Svyatkovskiy, A., Deng, S.K. and Sundaresan, N. 2021. Unit Test Case Generation with Transformers and Focal Context. arXiv:2009.05617. Retrieved from http://arxiv.org/abs/2009.05617.

[17] Wang, J., Huang, Y., Chen, C., Liu, Z., Wang, S. and Wang, Q. 2023. Software Testing with Large Language Model: Survey, Landscape, and Vision. arXiv:2307.07221. Retrieved from http://arxiv.org/abs/2307.07221.