Exploring Assertion Generation via Large Language Model

Bowen Yu)
boy014@ucsd.edu

1 Introduction

Unit testing has become an indispensable practice in the software development lifecycle (Godefroid et al., 2005). It involves creating unit tests specifically designed to ensure that individual components—such as methods, classes, or modules—function according to their intended specifications and usage requirements.

Writing high-quality unit tests manually, however, is a challenging and time-intensive task. It demands sufficient testing expertise and a significant investment of time. For example, prior research (Daka and Fraser, 2014) indicates that developers typically spend over 15% of their time writing unit tests. To address this, substantial efforts have been directed towards the automated generation of unit tests (Fraser and Arcuri, 2011; Pacheco and Ernst, 2007). A unit test generally consists of a test prefix (i.e., a sequence of statements that prepare the unit under test to reach a specific state) and a test assertion (i.e., an oracle defining the conditions that should hold true in that state) (Dinella et al., 2022).

To improve assertion generation and supplement existing test generation tools, an increasing number of assertion generation (**AG**) techniques (Watson et al., 2020; Yu et al., 2022) have emerged. These techniques can be broadly categorized into three groups: deep learning (DL)-based, information retrieval (IR)-based, and integrated methods (referred to as *Integration*), which combine DL and IR approaches.

In particular, DL-based AG techniques (Watson et al., 2020) employ deep learning models to take a test prefix and its corresponding focal method (i.e., the method under test) as input and generate assertions from scratch[1]. Conversely, IR-based AG techniques (Yu et al., 2022) retrieve similar focal-tests from a corpus and adapt the retrieved assertions to fit the desired output. Integration techniques (Yu et al., 2022) aim to combine the strengths of both DL and IR approaches to produce suitable assertions.

Recently, large language models (LLMs) have garnered significant attention in the software engineering (SE) community, achieving remarkable performance across a wide range of code-related tasks. In this paper, I aim to address these gaps by exploring the potential of leveraging LLMs to generate unit test assertions and support future advancements in unit testing research. Below is the plan and its current completion status:

- Collected and preprocessed dataset: DONE.
- Utilized LLM on AG task and finetune: DONE
- Compare experimental results and analyze: DONE

2 Related work

In recent years, researchers have started exploring the effectiveness of deep learning approaches for generating test assertions based on a mass of historical developer-written test cases. These approaches utilize deep neural networks to learn from the existing test cases and generate assert statements. For example, Watson *et al.* (Watson et al., 2020) propose *ATLAS*, an approach that utilizes Neural Machine Translation (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014) to generate syntactically and semantically correct unit test assert statements for a given *focal-test*. A focal-test consists of a test prefix and its focal method (*i.e.,* the method under test). Further, inspired by the idea of combining Information Retrieval (IR) and Deep Learning (DL), Yu *et al.* (Yu

[1]For consistency with prior research (Watson et al., 2020), I refer to such input pairs as *focal-test*.

et al., 2022) propose a retrieval-augmented deep assertion generation method. They introduce two IR-based approaches, namely IR-based assertion retrieval (IR_{ar}) and retrieved-assertion adaptation (RA_{adapt}), for generating assertions. The IR_{ar} approach retrieves the assertion with the highest similarity to the given focal-test, using measures like Jaccard similarity. Although IR_{ar} may not always retrieve completely accurate assertions, RA_{adapt} automatically adapts the retrieved assertions by replacing tokens within them. Two strategies are proposed for determining replacement values: a heuristic-based approach (RA^{H}_{adapt}) and a neural network-based approach (RA^{NN}_{adapt}). In addition, Yu *et al.* (Yu et al., 2022) combine IR and DL techniques and propose an integrated approach, referred to as *Integration*. Based on *Integration*, Sun *et al.* (Sun et al., 2023) further propose a retrieval-augmented deep assertion generation method, namely EDITAS. EDITAS views the assertion from a similar focal-test as a prototype and leverages a neural sequence-to-sequence model to learn the assertion edit patterns used to modify the prototype.

3 Datasets

In this paper, I utilize two datasets, namely $Data_{old}$ and $Data_{new}$, which were introduced by Watson *et al.* (Watson et al., 2020) and Yu *et al.* (Yu et al., 2022), respectively. These datasets are publicly available and have been widely used in assertion generation research (Watson et al., 2020; Yu et al., 2022; Sun et al., 2023).

The $Data_{old}$ dataset is derived from the original dataset used by *ATLAS*. To construct $Data_{old}$, a pool of 2.5 million test methods was initially collected from GitHub, including test prefixes and their corresponding assertion statements. Each test method in $Data_{old}$ was concatenated with a focal method, which represents the production code under test. After applying preprocessing steps (detailed in Section 3.1), the dataset was reduced to 156,760 samples.

Although $Data_{old}$ provides a clean dataset, excluding assertions with unknown tokens oversimplifies the problem of generating assertions, potentially limiting its ability to represent real-world data distributions. To address this, Yu *et al.* (Yu et al., 2022) constructed an expanded dataset, $Data_{new}$, by including cases that were excluded from $Data_{old}$ due to the presence of unknown tokens. This addition introduced 108,660 samples, increasing the total size of $Data_{new}$ to 265,420 samples.

3.1 Preprocessing Steps

To ensure the quality and consistency of the datasets, I applied the following preprocessing steps:

1. **Token Length Filtering:** Test methods with token lengths exceeding 1K were excluded from the dataset. This step helps eliminate outliers and reduces noise caused by overly complex or lengthy test methods.

2. **Unknown Token Removal:** Assertions containing unknown tokens not present in the focal-test or the vocabulary were filtered out. While this step ensures consistency, it also introduces a limitation by potentially removing real-world cases, which was later addressed by $Data_{new}$.

3.2 Dataset Splitting

To facilitate experiments and ensure comparability with prior studies (Yu et al., 2022; Sun et al., 2023), I divided both $Data_{old}$ and $Data_{new}$ datasets into training, validation, and test sets using an 8:1:1 ratio.

4 Baselines

To address the above-mentioned research questions (RQs), I compare two large language models (LLMs) with six state-of-the-art assertion generation (AG) techniques. These techniques include sequence-to-sequence-based, retrieval-based, and retrieval-enhanced learning-based approaches.

First, I include *ATLAS*, the first and classical AG technique that employs a sequence-to-sequence model to generate assertions from scratch. *ATLAS* serves as the most relevant baseline since both *ATLAS* and the studied LLMs treat assertion generation as a neural machine translation task built on the encoder-decoder transformer architecture.

Second, I incorporate three existing retrieval-based AG techniques: IR_{ar}, RA^{H}_{adapt}, and RA^{NN}_{adapt}. These techniques leverage retrieval mechanisms to find and adapt relevant examples for generating assertions.

Finally, I consider one state-of-the-art retrieval-enhanced learning-based approach, *Integration*,

and its most recent follow-up, EDITAS. The experimental results of all baselines on the two datasets are derived from the results reported by Sun *et al.* (Sun et al., 2023).

5 Approach

Model Selection In this experiment, I choose two models, *i.e.,*, CodeBERT and CodeT5. **CodeBERT** (Feng et al., 2020) is a bi-modal LLM designed for both programming language and natural language understanding. It utilizes a hybrid objective function during pre-training, combining standard *masked language modeling* (MLM) (Devlin et al., 2019) and *replaced token detection* (Clark et al., 2020). **CodeT5** (Wang et al., 2021) adopts the T5 architectures and incorporates token-type information specific to the sourcecode. In additional to a common denoising pre-training task, CodeT5 further leverages the semantics conveyed by identifiers assigned by developers.

Experimental Design. To explore LLM's performance on AG, I evaluate the performance of two LLMs and six state-of-the-art AG techniques, including *ATLAS*, IR_{ar}, RA^{H}_{adapt}, RA^{NN}_{adapt}, *Integration*, and EDITAS. To ensure a fair comparison, I provide the same set of test inputs to all methods under evaluation and use the same training set to train/implement the AG methods or fine-tune the LLMs. Each method takes the focal-test as input and generates the corresponding assertion as output. All training and evaluation experiments are conducted on an Ubuntu 20.04 server equipped with four NVIDIA GeForce RTX 3090 GPUs. **Results.** Figure 1 and Figure 2 presents

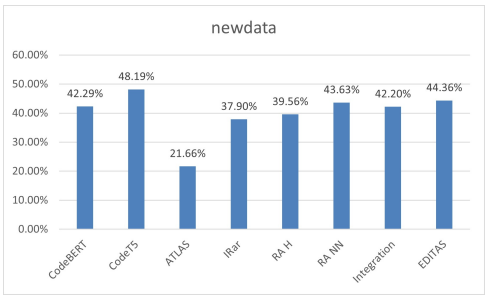

Figure 1: Data$_{new}$ dataset

the comparison results of LLMs and state-of-the-art AG approaches in terms of prediction accuracy. On average, LLMs achieve a prediction accuracy of 45.24% on the $Data_{new}$ dataset, outperforming baselines with an improvement of 21.25%. In particular, compared to the most relevant baseline

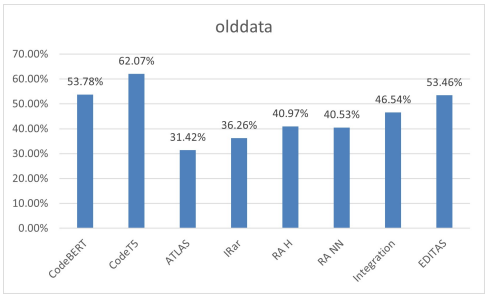

Figure 2: Data$_{old}$ dataset

ATLAS, LLMs achieve a prediction accuracy improvement of 109% on average. A similar trend is observed on the $Data_{old}$ dataset, where the improvement against other baseline methods is 39.48%. Since both LLMs and *ATLAS* frame the assertion generation problem as a neural machine translation task on an encoder-decoder architecture, these results underscore the advantages of LLMs over the default transformer in generating assertions.

Among the studied LLMs, CodeT5 consistently achieves the best performance. I attribute CodeT5's superior performance to its architecture and pre-training datasets. Specifically, CodeT5 employs an encoder-decoder architecture, which has proven effective in prior code generation tasks. Additionally, it is pre-trained on both CodeSearchNet and BigQuery datasets, which provide a larger corpus compared to other LLMs.

In contrast, encoder-only models such as CodeBERT require a separate decoder for generation tasks, starting from scratch and failing to leverage the benefits of pre-training. These results demonstrate the superior capability of CodeT5 in assertion generation.

6 Error analysis

CodeT5 demonstrates the best performance on both datasets, but its advantage varies across contexts. On the $Data_{new}$ dataset, it achieves 48.19%, outperforming CodeBERT by 5.9 percentage points. However, its margin over retrieval-enhanced methods like EDITAS narrows to just 3.83 percentage points, indicating the competitiveness of retrieval-based approaches on noisy datasets.

On the $Data_{old}$ dataset, CodeT5 achieves 62.07%, significantly outperforming CodeBERT by 8.29 percentage points and *ATLAS* by 30.65 percentage points. This suggests that CodeT5 ben-

efits from cleaner data due to its encoder-decoder architecture, while CodeBERT struggles with generation tasks due to its encoder-only design.

Overall, CodeT5 excels with its robust architecture, but retrieval-based methods perform competitively on noisier datasets, highlighting the impact of dataset characteristics and model architecture on assertion generation.

7 Conclusion

This project demonstrated the superiority of CodeT5 over CodeBERT and baseline methods for assertion generation, attributed to its encoder-decoder architecture and extensive pre-training. While retrieval-enhanced methods like EDITAS performed competitively on noisy datasets, CodeT5 excelled overall. The main challenge was ensuring fair comparisons across diverse methods and datasets. Future work could focus on hybrid approaches combining retrieval and LLM-based techniques to further enhance performance.

8 Acknowledgements

In this report, I used ChatGPT to revise my writings

References

Clark, K., Luong, M., Le, Q. V., and Manning, C. D. (2020). ELECTRA: pre-training text encoders as discriminators rather than generators. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Daka, E. and Fraser, G. (2014). A survey on unit testing practices and problems. In *25th IEEE International Symposium on Software Reliability Engineering, ISSRE 2014, Naples, Italy, November 3-6, 2014*, pages 201–211. IEEE Computer Society.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2019). BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Dinella, E., Ryan, G., Mytkowicz, T., and Lahiri, S. K. (2022). TOGA: A neural method for test oracle generation. In *44th IEEE/ACM 44th International Conference on Software Engineering, ICSE 2022, Pittsburgh, PA, USA, May 25-27, 2022*, pages 2130–2141. ACM.

Feng, Z., Guo, D., Tang, D., Duan, N., Feng, X., Gong, M., Shou, L., Qin, B., Liu, T., Jiang, D., and Zhou, M. (2020). Codebert: A pre-trained model for programming and natural languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 1536–1547. Association for Computational Linguistics.

Fraser, G. and Arcuri, A. (2011). Evosuite: automatic test suite generation for object-oriented software. In *SIGSOFT/FSE'11 19th ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE-19) and ESEC'11: 13th European Software Engineering Conference (ESEC-13), Szeged, Hungary, September 5-9, 2011*, pages 416–419. ACM.

Godefroid, P., Klarlund, N., and Sen, K. (2005). DART: directed automated random testing. In Sarkar, V. and Hall, M. W., editors, *Proceedings of the ACM SIGPLAN 2005 Conference on Programming Language Design and Implementation, Chicago, IL, USA, June 12-15, 2005*, pages 213–223. ACM.

Kalchbrenner, N. and Blunsom, P. (2013). Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1700–1709. ACL.

Pacheco, C. and Ernst, M. D. (2007). Randoop: feedback-directed random testing for java. In *Companion to the 22nd Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, OOPSLA 2007, October 21-25, 2007, Montreal, Quebec, Canada*, pages 815–816. ACM.

Sun, W., Li, H., Yan, M., Lei, Y., and Zhang, H. (2023). Revisiting and improving retrieval-augmented deep assertion generation. *CoRR*, abs/2309.10264.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Wang, Y., Wang, W., Joty, S., and Hoi, S. C. (2021). Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. *arXiv preprint arXiv:2109.00859*.

Watson, C., Tufano, M., Moran, K., Bavota, G., and Poshyvanyk, D. (2020). On learning meaningful assert statements for unit test cases. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, ICSE '20, page 1398–1409, New York, NY, USA. Association for Computing Machinery.

Yu, H., Lou, Y., Sun, K., Ran, D., Xie, T., Hao, D., Li, Y., Li, G., and Wang, Q. (2022). Automated assertion generation via information retrieval and its integration with deep learning. In *Proceedings of the 44th International Conference on Software Engineering*, ICSE '22, page 163–174, New York, NY, USA. Association for Computing Machinery.