Exploring the Use of Natural Language Processing Techniques for Enhancing Genetic Improvement

Oliver Krauss
Advanced Information Systems and Technology - AIST
University of Applied Sciences Upper Austria
Hagenberg, Upper Austria, Austria
ORCID - 0000-0002-8136-2606

***Abstract*—We explore the potential of using large-scale Natural Language Processing (NLP) models, such as GPT-3, for enhancing genetic improvement in software development. These models have previously been used to automatically find bugs, or improve software. We propose utilizing these models as a novel mutator, as well as for explaining the patches generated by genetic improvement algorithms. Our initial findings indicate promising results, but further research is needed to determine the scalability and applicability of this approach across different programming languages.**

***Index Terms*—genetic improvement, artificial intelligence, natural language processing, non-functional properties**

I. Introduction

We propose the integration of large-scale natural language processing (NLP) models into the genetic improvement (GI) process for software improvement, such as finding bugs or improving non-functional properties. These NLP models can be used as novel genetic operators, as well as for explaining the patches generated via GI.

The use of NLP models has been demonstrated in code generation [1], program repair [2] and optimization [3]. Recently, large-scale models have been specifically trained for understanding and generating source code, such as CodeT5 [4] and Codex [5]. ChatGPT, further advances the explainability of source code [6]. Our proposed approach aims to leverage the capabilities of these models to enhance the efficiency and understanding of the GI process.

The application of NLP models in software development has been shown to be effective in repairing bugs and improving programs on its own [4]. However, their integration with GI techniques can provide even greater benefits. In this paper, we propose using NLP models in the following ways:

- As genetic operators: NLP models can be used to perform different mutation operations, providing more options for generating new solutions.
- Simplification of patches: NLP models can be used to simplify larger patches generated by the GI algorithm, making them easier to understand and inject into an existing code base.
- Explanation of patches: NLP models can be used to explain the reasoning behind the patches generated by the GI algorithm, providing greater insight into the solution and increasing the transparency of the patch.

As an example we use the code provided in Listing 1, taken from a publication by Licorish and Wagner in a previous GI workshop [7]. This example serves to demonstrate the potential of using NLP models in conjunction with GI techniques.

Listing 1. Transformation of int array to Integer array. Originally listed as Listing 6 in [7]. The mutation on line 5 (yellow background) of the original code was produced with the conventional GI *copy* mutator.

```
public class C315640{
  private static Integer[]
     toIntegerArray(int[] array){
    Integer[] finalArray = new
       Integer[array.length];
    for (int i=0; i<array.length; i++){
      finalArray[i] = array[i];
      finalArray[i] = array[i];
    }
    return finalArray;
  }
}
```

II. Utilization of NLP Models as Genetic Operators in GI

The integration of NLP models into the GI process can provide significant benefits. These models can be used as powerful genetic operators, capable of automating the rewriting of code for specific goals, repairing bugs, and implementing safety checks or documentation.

As an example, when using an NLP model to improve the code in Listing 1 and add Java documentation, the results are impressive. It simplifies the code from a for loop to a single line, includes null checking, and documents the function. This can be observed in Listing 2. The return statement encapsulates the core functionality of the original code, while the remaining code serves as an enhancement provided by the NLP model.

For example when asked to improve the code in and add javadoc the result is rather impressive, as it not only simplifies the code from a for loop to a single line, but also adds safety measures for nullchecking, and correctly documents the given function. This can be seen in Listing 2.

Another potential application of NLP in GI is as a form of selection operator. By providing the NLP model with the individuals in the population, it can be used to select distinct or viable solutions.

979-8-3503-1232-4/23/$31.00 ©2023 IEEE
DOI 10.1109/GI59320.2023.00014

Listing 2. Improved code via ChatGPT

```
/**
* Converts an array of primitive ints to
    an array of wrapper class Integers
*
* @param array the array of primitive
    ints to be converted
* @return an array of wrapper class
    Integers
* @throws IllegalArgumentException if
    the input array is null or empty
*/
private static Integer[]
    toIntegerArray(int[] array) {
    if (array == null || array.length ==
        0) {
        throw new
            IllegalArgumentException("Input
            array cannot be null or
            empty");
    }
    return Arrays.stream(array)
        .boxed().toArray(Integer[]::new);
}
```

In cases where the GI algorithm generates larger patches, NLP models can be used to reduce code bloat [8] by detecting duplicates and simplifying the code through refactoring. This approach can also be used to improve and document the final patch generated by the GI algorithm, as demonstrated in Listing 2.

III. Explaining Generated Patches using Natural Language Processing

NLP can be used to explain the patches generated by GI or to simplify them after the GI run has completed. For instance, when using a model like ChatGPT to explain line 5 in Listing 1, it provides the following response:

> Line 5 assigns the value of the int at the current index of the input array (array[i]) to the corresponding index of the finalArray (finalArray[i]). It is duplicated in the code which will have no effect on the result.

This not only explains the patch that was created, but also informs the user that the patch will not have any effect on the source code. Initial testing has revealed that models such as ChatGPT can effectively explain source code, even identifying core concepts like the use of the Newton-Raphson approximation method in mathematical functions, without relying on code comments or the original function names. However, smaller models like CodeT5 and Codex may provide nonsensical answers and struggle to explain code that deviates from standard algorithms that they may have inferred from recurring sample projects or Stack Overflow articles.

IV. Challenges and Future Directions in NLP-assisted GI

The integration of NLP in GI poses two main challenges: cost and applicability. NLP models can be quite large. Flan-T5-xl, has 68GB of files and requires a significant amount of RAM. Paid APIs such as GPT-3 can be used, but these can become cost-inefficient when used for operations that require several hundreds or thousands of API calls. The applicability of NLP-assisted GI need to be evaluated across different programming languages and domains. For instance, CodeT5 performs well on Python code, and ChatGPT and Codex are effective for C and Java code. Their ability to understand and explain code in other languages may be less robust.

GI code is often less human-readable than the code written by a developer, which may impact NLP models. NLP models rely on the naming of variables and functions for understanding code, for example when renaming all names in the inverse square root function, ChatGPT incorrectly assumed that the code calculates 1/x, but still found the connection to the Newton Raphson approximation method.

As a next step, we plan to design prompts for the NLP models that can be used as genetic operators and for processing the final results of the GI algorithm. These prompts will be made available in a publicly hosted repository, together with a study that shows the application of NLP in GI on program repair and non functional optimization. Additionally, we aim to conduct a comprehensive comparison of currently available models in terms of the quality of the results they generate and the monetary or processing costs associated with their integration into the GI process. This will provide insights into cost-effective and efficient options for NLP-assisted GI.

An exploration of complementing NLP models with GI would also be interesting. GI can be used to optimize the prompts of NLP models via its fitness function, or fix nearly perfect solutions produced by NLP.

References

[1] M. Zong and B. Krishnamachari, "a survey on gpt-3," 2022. [Online]. Available: https://arxiv.org/abs/2212.00857

[2] C. S. Xia and L. Zhang, "Less training, more repairing please: Revisiting automated program repair via zero-shot learning," 2022.

[3] S. Garg, R. Z. Moghaddam, C. B. Clement, N. Sundaresan, and C. Wu, "Deepperf: A deep learning-based approach for improving software performance," 2022.

[4] Y. Wang, W. Wang, S. R. Joty, and S. C. H. Hoi, "Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation," *CoRR*, vol. abs/2109.00859, 2021. [Online]. Available: https://arxiv.org/abs/2109.00859

[5] M. C. et al., "Evaluating large language models trained on code," *CoRR*, vol. abs/2107.03374, 2021. [Online]. Available: https://arxiv.org/abs/2107.03374

[6] T. B. B. et al., "Language models are few-shot learners," *CoRR*, vol. abs/2005.14165, 2020. [Online]. Available: https://arxiv.org/abs/2005.14165

[7] S. A. Licorish and M. Wagner, "Dissecting copy/delete/replace/swap mutations: Insights from a gin case study," in *GECCO 2022*. New York, NY, USA: Association for Computing Machinery, 2022, p. 19401945. [Online]. Available: https://doi.org/10.1145/3520304.3533970

[8] S. Luke and L. Panait, "A Comparison of Bloat Control Methods for Genetic Programming," *Evolutionary Computation*, vol. 14, no. 3, pp. 309–344, 09 2006. [Online]. Available: https://doi.org/10.1162/evco.2006.14.3.309