# #157 How robust is neural code completion to cosmetic variance?



Email notification
Select to receive email on updates to reviews and comments.

PC conflicts
Christoph Treude Shane McIntosh

**Rejected**

Submission (234kB) 15 Oct 2021 11:53:46pm AoE · 3589ed2e

## ▼ Abstract

Neural language models hold great promise as tools for computer-aided programming, but questions remain over their reliability and the consequences of overreliance. In the domain of natural language, prior work has revealed these models can be sensitive to naturally-occurring variance and malfunction in unpredictable ways. A closer examination of neural language models is needed to understand their behavior in programming-related tasks. In this work, we develop a methodology for systematically evaluating neural code completion models using common source code transformations such as synonymous renaming, intermediate logging, and independent statement reordering. Applying these synthetic transformations to a dataset of handwritten code snippets, we evaluate three SoTA models, CodeBERT, GraphCodeBERT and RobertA-Java, which exhibit varying degrees of robustness to cosmetic variance. Our approach is implemented and released as a modular and extensible toolkit for evaluating code-based neural language models.

## ▶ Authors (blind)

B. Considine, J. Guo, X. Si [details]

## ▶ Topics

| | OveMer |
|---|---|
| Review #157A | 2 |

| | OveMer |
|---|---|
| Review #157B | 1 |
| Review #157C | 3 |

You are an **author** of this submission.

**Edit submission**

Reviews in plain text

## Review #157A

### Overall merit

**2.** Weak reject

### Paper summary

This paper proposes the use of semantic-preserving (cosmetic) transformations to evaluate the robustness of code-based neural language models. Experiments were conducted with three state-of-the-art neural models (RoBERTa, CodeBERT, and GraphCodeBERT), various transformations, and two usage scenarios. Results show that all models are sensitive to cosmetic transformations. GraphCodeBERT manifested less variance among the three models.

### Comments for author

- Value: The problem is worth exploring; it is important to better understand the limitations of code-based neural models. Considering novelty, I am not aware of applications of metamorphic testing to evaluate the robustness of neural models on those tasks. However, I highlight that MT has been used in many application scenarios. So, I don't see a big leap forward.

- Impact: The approach is simple and could be generally applied (without changes). So, the potential of making an impact on how neural models (trained for code-related tasks) are evaluated is high. Considering experiments, results are intriguing to me---if there is high variation in results with the cosmetic transformations, I result on the original code was not obtained by chance.

- Soundness: I failed to observe serious problems related to the technical correctness and plausibility of the experiments.

- Quality: Presentation needs to be improved.

==== higher-level presentation comments ====

- why the use of models as "extensional function" is relevant?

- explain consequences of observations on intro
- unclear how "permute argument order" is cosmetic (semantic-preserving). First, I thought this was for Python (which "named parameters") but then realized this is Java. Please, explain how permuting argument order is cosmetic.
- Can you justify why the code completion task is realized on "one token" only?

==== lower-level presentation comments ====

- define SoTA
- "adversial"
- "is treats"
- define SCT
- "alternate topological sort on the dataflow graph" is this really necessary?
- across "five different SCTs". shouldn't be four?
- histo*r*gram

## Review #157B

### Overall merit

**1.** Reject

### Paper summary

In this paper, the authors performed an empirical study on the robustness of various pre-trained code representation models. The authors proposed several common source code transformations such as synonymous renaming, intermediate logging, and independent statement reordering. Experiments on three SOTA models showed various degrees of robustness to cosmetic variance.

### Comments for author

Pros.

+ Interesting idea

Cons.

- Bias in the experiment setting
- It is not clear why the authors select the complexity to group the data
- Some part of the paper is hard to read

Overall, the idea is worth exploring and is novel, and the problem is relevant to software engineering research. However, I have several concerns about the experiments:

1. From my understanding, the authors only transform the source code in the testing set, if I understand correctly. However, I think it is biased if only the source code in the testing set is transformed, considering some tokens that appeared in the training set might not appear in the testing set due to the transformation. That is to say, I recommend the authors perform the following experiments:

Group 1: Original large-scale training set for pre-training + Original testing set

Group 2: Original large-scale training set for pre-training + Transformed testing set

Group 3: Transformed large-scale training set for pre-training + Original testing set

Group 4: Transformed large-scale training set for pre-training + Transformed testing set

Group 5: Original&Transformed large-scale training set for pre-training + Original testing set

Group 6: Original&Transformed large-scale training set for pre-training + Transformed testing set

By doing these experiments, we can clearly see how the difference in performance is introduced.

2. Currently, the authors used Dyck-1 complexity to group the data. However, it is not clear why the authors select the Dyck-1 complexity. Why did not the authors consider other metrics, such as the size of code?

3. The tables are hard to read, considering there is no title for all of the tables in the paper. Also, can the authors make the table simple by visualizing the results?

4. The writing of the paper should be improved. If I understand correctly, the authors performed two tasks, code completion and comment generation. If so, can the authors make it clear in the title and the body of the paper?

5. Lines 246-260 are hard to understand. Can the authors rephrase the paragraph to make it clear?

6. How do the three SOTA approaches work for code completion and comment generation? I am sorry I can only see the pre-trained models, but I do not know how the authors solve the two tasks.

## Review #157C

**Overall merit**

**3.** Weak accept

### Paper summary

This paper studies the robustness of existing neural code (language) models in the presence of variations such as independent statement reordering, synonymous renaming, and intermediate logging. The authors call these variations "cosmetic" in the sense that they do not change program behavior and are syntactically valid. The paper studies three different models with that goal. CodeBERT, RobertA-Java, and GraphCodeBERT. The authors find out existing models exhibit sensitivity even for this constrained type of transformation.

### Comments for author

I like this paper. I believe investigating whether code completion based on existing programming language models is sensitive to small cosmetic variations is **valuable** considering the current landscape of intensive research on how machine learning approaches can support software engineering activities. In particular, this is a nice paper for a NIER track because its results raise questions that can be addressed by future research: (i) how can we improve existing models to be able to deal with this kind of variation? (ii) how would these models be affected by compositions of these variations? (iii) what about more complex variations that change program semantics? In this sense, I think this is a potentially **impactful** paper. The proposed approach is mostly **sound**, in the sense that the evaluation metrics make sense and the methodology is mostly well-explained. There are a few points that I would like to see discussed better in the final version of the paper (see below). Finally, I think the paper is well-written and generally easy to follow.

There are some points that I would like to see clarified or better discussed in the final version of the paper. The first one pertains to the use of REGEXes. Although the authors acknowledge the limitations of using REGEXes, I still think this is not sufficient. Basically, they are using this approach to perform refactorings, something that some would see as overly optimistic in practice, in particular for renaming. I suggest the authors either discuss cases where this approach would not work, with a couple of examples, or provide actual data about the accuracy of this approach. The space currently dedicated to discussing alternatives that the paper does not use would, IMHO, be used better in this manner. An inaccurate approach to perform the transformations can put all the results of the paper in check.

I also suggest the authors better explain where do the samples used in the experiments come from. The paper just mentions that *"Our dataset consists of a hundred of the highest-starred Java repositories hosted by GitHub organizations with over 100 forks and between 1 and 10 MB in size"*. How does that translate into 704 snippets used to evaluate permuteArgument in the 1-20 complexity range?

Finally, it would be nice to have a little more discussion about the results. We have the numbers, but what do they mean? Is the sensitivity to cosmetic variance high or low? In many cases, the variance is greater than the statistic of interest. How does that impact our interpretation of the results? Are they reliable?

## Additional comments

- three SoTA models => what is a SoTA model?

- Our work is treats the model
- Many references are incomplete, not even mentioning the place of publication.
- SCT => source code transformation.
- (3) and (4) may produce semantic variants, strictly speaking, we cannot rule out the possibility that any of the aforementioned SCTs are semantically-preserving => This is a bit strange. I think the authors mean "semantics-altering". From my understanding, they expect the transformations to be, in general, semantics-preserving but cannot rule out the possibility that they occasionally alter program semantics.
- code analysis, however we have => code analysis. However we have
- masked token completion accuracy for code completion and ROUGE-synonym score for document synthesis. => I am not familiar with these metrics. I suggest the authors briefly explain them.
- reported in § 3. => a bit strange. I suggest "reported in Section 3".
- As we can see, RoBERTa is considerably more sensitive to cosmetic variance than CodeBERT and GraphCodeBERT, which are relatively close contenders. => I am not a specialist in the topic of this paper and this is not obvious to me. It is clear that RoBERTa is the worst for permuteArgument. However, for swapMultilineNo, it beats GraphCodeBERT in 4 out of the 9 rows.
- Why are the sample sizes in the second column of page 3 consistently higher for RoBERTa and consistently lower for GraphCodeBERT? What does that mean?
- as reported by 333 prior literature => [REF] missing.
- What is Figure 1 showing? The text does not refer to it. I suggest the authors add some text to explain what it is showing. Furthermore, I'd like to ask the authors why are the results for RoBERTa so low for swapMultilineNo and addExtraLogging? I assume it is because of the outliers that can be observed for each case. Considering this observation (in case it is true), I suggest the authors employ the median, instead of the average, to report the results in Figure 1, since the median is insensitive to outliers. If I understand the figure correctly, the use of the mean is giving the wrong impression about the results (that RoBERTa outperformed the two other models).
- depsite their
- seqeunce prediction