← Back to **Author Console** (/group?id=ICLR.cc/2022/Workshop/DL4C/Authors#your-submissions)

How Robust are Neural Code Completion Models to Source Code Transformation?

(/pdf?id=Sdbgl3_bJZ9)

Anonymous

03 Mar 2022 ICLR 2022 Workshop DL4C Blind Submission Readers: Everyone Show

Bibtex

Keywords: code completion, refactoring, robustness, testing, evaluation, self-supervision

TL;DR: We study the effect of source code transformations on the robustness of neural code completion models.

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 more methodical examination is necessary to understand their behavior on programming-related tasks. In this work, we develop a methodology for systematically evaluating neural code completion models using common source code transformations. We measure the distributional shift induced by applying those transformations to a dataset of handwritten code fragments on four pretrained models, which exhibit varying degrees of robustness under transformation. Preliminary results from those experiments and observations from a qualitative analysis suggest that while these models are promising, they should not be relied upon uncritically. Our analysis provides insights into the strengths and weaknesses of different models, and serves as a foundation for future work towards improving the accuracy and robustness of neural code completion.

Revealed to Breandan Considine, Xiaojie Xu, Xujie Si, jguo@cs.mcgill.ca

26 Feb 2022 (modified: 26 Feb 2022) ICLR 2022 Workshop DL4C Submission

Authors: *Breandan Considine (/profile?id=~Breandan_Considine2), Xiaojie Xu (/profile?id=~Xiaojie_Xu2), Xujie Si (/profile?id=~Xujie_Si1), Jin Guo (/profile?email=jguo%40cs.mcgill.ca)*

Add **Withdraw**

Reply Type: all Author: everybody Visible To: all readers **4 Replies**

Hidden From: nobody

[−] **Paper Decision**

ICLR 2022 Workshop DL4C Program Chairs

25 Mar 2022 ICLR 2022 Workshop DL4C Paper28 Decision Readers: Program Chairs,

Paper28 Authors

Decision: Reject

Comment: The authors present an analysis of the robustness of neural code models under semantics-preserving, cosmetic perturbations of the input, such as variable renaming and reordering of function arguments. Arguably, for many downstream tasks, the prediction should be unaffected by cosmetic transformations, but the authors find that they are not.

The reviewers are mostly satisfied with the presentation, but struggle to see the work's significance due to the lack of actionable takeaways. There are also concerns that the central premise of the paper is not sound, since the proposed perturbations may not be semantics-preserving or are transformations a model for code should be sensitive to. Due to these criticisms, we have decided to reject the paper.

[−] **Review**

ICLR 2022 Workshop DL4C Paper28 Reviewer efQn

19 Mar 2022 ICLR 2022 Workshop DL4C Paper28 Official Review Readers: Program Chairs, Paper28 Reviewer efQn, Paper28 Authors

Review:

Summary

This paper studies how various code language models (GraphCodeBERT, CodeBERT, RoBERTa-Java) change their results when making small perturbations to their inputs. In particular, the authors study renaming variable names with synonyms, adding logging statements, reordering some statements, and reordering arguments to functions. The authors argue that such changes should not affect the result of the model because they have very little effect on the semantics of the code.

Strengths

- The paper addresses an interesting problem about the robustness of code LMs when faced with variations with little effect on the programs' semantics.
- The authors provide an empirical analysis using multiple LMs and tasks.

Weaknesses

- Most of the paper is spent on background, motivation, and methods, and relatively little on the actual experimental results.
- The experiments don't directly measure how much a change to the input changes the result; rather they study how the accuracy against fixed labels changes. This is a roundabout way of measuring the effect of perturbations.
- It is arguable whether the actual perturbations used should indeed be ones that the models should pay no attention to. For example, permuting argument orders may contravene common conventions used for argument order. Renaming variables with synonyms may cause the use of names that are not commonly used in programming, or end up picking a new word with different connotations (even if it has some overlap in meaning with the original word). Swapping "independent" lines (two lines which share no tokens) may significantly change the meaning of the code, depending on the side effects of the lines.

Suggestions

- Figure 1 should be reformatted. For example, it should be split into three sections and made much larger so that it's possible to read the text. I believe the before/after comparisons can also be made clearer than it is now.
- It should be possible to train models which are invariant to these kinds of perturbations, by design. For example, the models can be blind to specific names used for variables (they can be all anonymized before being given to the model), lack access to the ordering of "independent" lines in function bodies, and be invariant to function argument ordering. It would be interesting to see if such models do worse in real-world cases due to the smaller amount of information they get compared to existing code LMs which use the code directly.

Rating: 5: Marginally below acceptance threshold

Confidence: 4: The reviewer is confident but not absolutely certain that the evaluation is correct

[−]

Experiments seem fine, but results are unconvincing and no actionable takeaways.

ICLR 2022 Workshop DL4C Paper28 Reviewer 1uNa

17 Mar 2022 ICLR 2022 Workshop DL4C Paper28 Official Review Readers: Program Chairs, Paper28 Reviewer 1uNa, Paper28 Authors

Review:
Quality: The paper raises some valid concerns and processes a set of experiments to investigate the validity of these concerns. The authors provide code through an anonymous github, which inspires confidence in the results. Overall, the experiments seem fine but the results are not convincing.

Clarity: Some missing details hurt the presentation. For example, I couldn't find a clear explanation of what the (before) and (after) suffixes meant. Moreover, the graphs themselves were way too small to be read carefully. I would suggest breaking up Figure 1 into multiple figures.

Originality: To my knowledge, this type of analysis has not really been studied for code models, though similar type of robustness analyses have been explored in the natural language space.

Significance: I think the authors should spend more time explaining to the readers why the results have significance. I applaud them for adding some conclusions in bullet points towards the end, but some further discussion would help enlighten the reader. It would be nice to test the performance of causal language models to see if the same story holds in that setting.

Pros:

- Considers a diverse range of models.

Cons:

- Some more discussion on the results would help the presentation.
- Pictures should be bigger.
- No actionable takeaways. While the authors present their conclusions on page 5, it's not clear what a practitioner or researcher should take away from them.

Rating: 5: Marginally below acceptance threshold
Confidence: 3: The reviewer is fairly confident that the evaluation is correct

[–]
Official Review of "How Robust are Neural Code Completion Models to Source Code Transformation?"

ICLR 2022 Workshop DL4C Paper28 Reviewer AssG

16 Mar 2022 ICLR 2022 Workshop DL4C Paper28 Official Review Readers: Program Chairs, Paper28 Reviewer AssG, Paper28 Authors

Review:
Summary: This paper explores how sensitive code language models (mostly BERT variants) trained on code are to cosmetic changes in the underlying code (which in theory preserve semantics). The paper finds that almost all models are quite sensitive to cosmetic changes like variable renaming, calling into question how effectively they can replace traditional forms of static analysis.

Comments:

- Overall the paper is excellent. It's concise and answers a clearly stated question in a clear way.
- The experimental code is released and open-sourced.
- I'd like to see more example of rewritten source code in practice. Looking at the few examples in the appendix, it seems like for instance, the variable renaming is more aggressive than the text of the paper would suggest, e.g. renaming "buffer" to "cushion" via a mistaken synonym. The actual magnitude of the change seems critical to how "reasonable" the model behavior is. It would be good to have an additional experiment which varies how aggressive these changes are.

- While this paper presents its results clearly, I'm not quite sure what to take away from it. For static analysis tasks, it's clear the model should be robust to these kinds of syntactically irrelevant changes, but for things like code completion, even traditional non-DL models use variable names to guide completion. It would make more sense in my opinion to explore static analysis tasks related to e.g. variable misuse or other syntactic or semantic errors.
- Overall, this is an excellent paper!

Rating: 6: Marginally above acceptance threshold

Confidence: 5: The reviewer is absolutely certain that the evaluation is correct and very familiar with the relevant literature