An Empirical Analysis of InCoder on the Statement Prediction Task

Frank van der Heijden
f.n.m.vanderheijden@student.tudelft.nl

Supervisor: Maliheh Izadi
Advisor: Georgios Gousios
Professor: Arie van Deursen

01 Introduction

Automatic code completions can be made by various code language models, and these can be differentiated in three categories: single token completion, statement (line) completion and block completions.

Automatic code completions for developers (e.g. IntelliSense) usually happen on trigger points, which are pre-defined points in the syntax of the code, and a similar approach can be used for statement predictions [1].

In this study we evaluate the InCoder [2] model on the statement prediction task, using SOTA metrics.

02 Methodology

For the evaluation, two novel datasets have been created: P1K-22 and JS1K-22, based on the top-1000 most-starred GitHub repositories.

Inferencing is then done on two versions of these datasets: the raw files and a version without comments.

Additionally, test cases on trigger points are filtered from these datasets and compared.

Finally, we calculate the prediction scores using the SOTA metrics EM, Edit Similarity, BLEU-4, ROUGE-L F1, and METEOR.

286.705 Source Code Files
4.818.440 Total Test Cases
- **1.040.938** Test Cases in P1K-22
- **1.368.282** Test Cases in JS1K-22

TU Delft Delft University of Technology

03 Results

On average, an improvement of 9.9% on Exact Match was achieved when using both contexts, and similar results for the other metrics, as can be seen in figure I.

Interestingly, for Python, statement predictions on both contexts were considerably better without comments, and on the left context lower on exact match, but similar on the other metrics (figure II).

On a similar note, JavaScript seems to perform very well on trigger points, but performs slightly worse on code without comments (figure III).

04 Conclusion

In conclusion, using both contexts produce significantly better results in all cases: on the raw dataset, without comments, and on trigger points.

Furthermore, without comments, the P1K-22 dataset produced better results.

Finally, trigger points show to be promising in both datasets, and perform even better on the JS1K-22 dataset.

For future research, the benefit of using contexts across multiple documents could be researched. However, the model needs to be changed to handle this kind of input.

Figure I

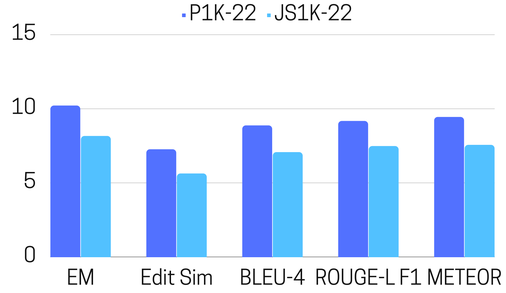


Figure II

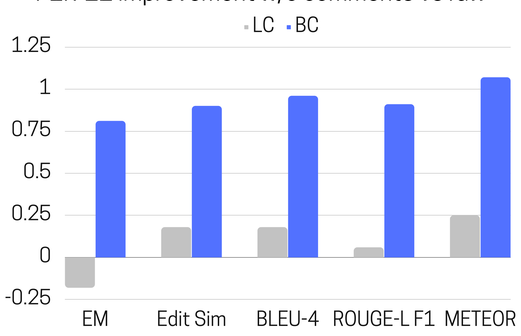
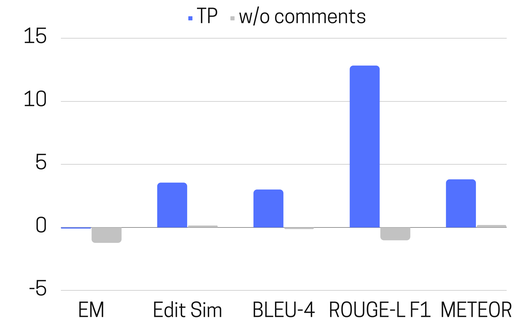


Figure III



Related literature

[1] Izadi et al. (2022). CodeFill: Multi-token Code Completion by Jointly Learning from Structure and Naming Sequences. arXiv preprint arXiv:2202.06689
[2] Fried et al. "Incoder: A generative model for code infilling and synthesis." arXiv preprint arXiv:2204.05999(2022).