Critique Report on the Paper: “Incorporating External Knowledge through Pre-training for Natural Language to Code Generation [2]”

Sumaiya Tabassum Nimi

Background Information and Purpose of the Research

Manually labelled dataset are often inadequate for training deep learning models for automatic code generation. Since developers typically resort to online question answer platform called StackOverflow and API documentation of the programming language, the paper [2] adopted a similar route for training code generator models, using NL-code pairs obtained from these two sources.

Methods Proposed

Training of the deep learning model for code generation was done in the following two steps.

- Pre-training: In this phase the model was pre-trained on specification-code pairs collected from following two repositories other than the available labelled dataset.
  - StackOverflow: Pairs of NL-code collected from StackOverflow often tend to be irrelevant. These collected pairs were ranked using [3], reflecting the worth of the pairs.
  - API Documentation: Codes collected from API documentation are always precise, but their descriptions tend to be lengthy. Hence several heuristics were applied to extract the NL intents out of them. Also, frequencies of API entries for different libraries might not always reflect frequencies of their real usages, unlike StackOverflow, and hence might induce adverse bias towards libraries having higher number of entries. In order to resolve this issue, collected data from API were re-sampled using a retrieval based method proposed in [1], through exploiting the frequencies of the NL-code pairs obtained both from manually annotated dataset and mined data obtained from StackOverflow, that reflect the real-life usage statistics.
- Fine-tuning: After pre-training, the resulting model was fine-tuned using available manually curated dataset containing clean data samples.

Strength of the Work

- Novel incorporation of freely and widely available, as well as the developers' favorite StackOverflow and API codes for pre-training of code generators.
- Re-sampling done on the API data based on real usage statistics and hypothesis reranking [5] led to obtaining state-of-the-art performance on benchmark CoNaLa [3] testbed in terms of BLEU score.
- Some case studies reported in the paper also qualitatively validated that the proposed method could lead to more accurate codes, in terms of calls to the pertinent API functions with more correct positioning of the arguments.

Limitations of the Work

- The experimental settings seemed very limited. The proposed training approach was applied to train the TranX [4] model only. It would have been interesting to analyze the performance of some other popular models for code generation on application of the proposed training method. There is a possibility that better performance would have been obtained for those models. The results should have been reported to show at least that this was not the case.
- The results were reported in terms of BLEU score only. While these might be arguably the best metric for evaluation for these categories of problems, reporting results on other metrics could have further solidified the effectiveness of the proposed strategy.
- Ablation studies regrading effects of the different values of the hyperparameters like k and τ should have been done, other than reporting results on some fixed values of these hyperparameters. Some visualizations at least should have been included to show how the performance was affected by choice of the hyperparameters.
- At least some of the related works should have been discussed in at least summarized version in the paper. It is always better to assume that not all the readers are familiar with the domain and a general discussion should be there so that such readers can grasp the significance of the work from the perspective of related researches conducted in the particular field.

Questions Unanswered

- It was not clear just by reading the paper how the NL inputs and pieces of source codes were collected from StackOverflow data. As both the questions and answers StackOverflow contain many unnecessary verbose descriptions often, some schematics should have been shown in the paper about how the NL-code pairs were generated out of the questions and answers.
- The discussion about the heuristics used to extract pertinent NL intent out of lengthy API documentation provided in the paper were not very objective enough for reproduction. These heuristics should have been discussed with proper use cases and some visualizations like flow-charts.

Suggested Future Studies

- The proposed training approach could be further improved with inclusion of source codes from GitHub, one of the most popular repositories for academic and industrial purposes, as a source of external data for pre-training. The NL intents could be collected from readme files of the repositories and the code snippets are already there beautifully arranged in a directory-like structure.

References

[1] K Spark-Jones, S Walker, and SE Robertson. A probabilistic model of information retrieval: Development and comparative experiments-part 1 and 2. *Information Processing and Management*, 36(6):779–840, 2000.

[2] Frank F Xu, Zhengbao Jiang, Pengcheng Yin, Bogdan Vasilescu, and Graham Neubig. Incorporating external knowledge through pre-training for natural language to code generation. *arXiv preprint arXiv:2004.09015*, 2020.

[3] Pengcheng Yin, Bowen Deng, Edgar Chen, Bogdan Vasilescu, and Graham Neubig. Learning to mine aligned code and natural language pairs from stack overflow. In *2018 IEEE/ACM 15th International Conference on Mining Software Repositories (MSR)*, pages 476–486. IEEE, 2018.

[4] Pengcheng Yin and Graham Neubig. Tranx: A transition-based neural abstract syntax parser for semantic parsing and code generation. *arXiv preprint arXiv:1810.02720*, 2018.

[5] Pengcheng Yin and Graham Neubig. Reranking for neural semantic parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4553–4559, 2019.