Exploring Existing Machine Learning Approaches Discerning the Quality of Source Code Identifiers

Marcel Simader
AI for Software Engineering, Topic 3
k11823075 / SKZ 521
marcel.simader@jku.at

Melissa Frischherz
AI for Software Engineering, Topic 3
k12011649 / SKZ 521
melissa.frischherz@gmail.com

***Abstract*—The quality of source code identifiers has a direct, and measurable impact on program comprehension, indirectly influencing validity, maintainability, and security of software. This paper proposes a small-scale survey of existing approaches assessing the quality of identifiers. Such a measure facilitates the evaluation of naming conventions in big code bases, suggestions of context-aware variable, method, or type names, or even finding semantic relationships between identifiers across languages and programs.**

By leveraging promising advancements in natural language processing (NLP) and machine learning, particularly the recent large language models (LLM), to analyze the information found in identifiers, we could greatly improve the code analyst's toolkit.

***Index Terms*—Machine Learning, Natural Language Processing, Code design, Maintainability, Software Quality/SQA.**

I. Introduction

Identifiers are the smallest unit of semantic information in program source code, yet they make up the majority of it [1]. This makes them both important to developers, who need them to comprehend software systems, and practically indispensable to nearly all analytical tools [2]. An empirical study published by Butler et al. shows that making poor choices for method, class, and type names correlates with less readable, maintainable, and overall worse quality code [1].

In the following sections, we present a survey of various papers with a focus on identifiers in the context of software engineering and artificial intelligence. We pay special attention to papers that attempt to qualitatively assess identifier names, which opens the path to generating context-dependant identifier suggestions, or finding failures to adhere to (implicitly) established naming conventions. Section II will give some background on the theoretical and technical frameworks, section III will discuss how we searched for, and selected papers, section IV will present the findings of the survey, and finally sections V and VI will conclude it with a discussion and ideas for future work.

II. Background

A. Machine Learning

***TODO*: Brief discussions of supervised/unsupervised techniques, neural networks, deep learning, and RNNs.**

B. Natural Language Processing

***TODO*: Brief discussions of n-grams, grammar, and the attention mechanism.**

***TODO*: EXPLAIN OOV.**

III. Methodology

We started by establishing an initial overview of the topic landscape, which led us to a more systematic search of papers pertaining to the ideas discussed in Section I. This was done by crafting search strings with relevant keywords for IEEEXplore[1] and Google Scholar[2]. The search queries we used for both services are as follows:

IEEEXplore: "(AI OR LLM OR Language Model OR NLP) AND Identifiers AND (Completion OR Analysis OR Suggestion)"

Google Scholar: "Large Language Models for measuring quality of identifiers in source code"

Of the 190 papers the IEEEXplore query yielded, we hand selected 30 to include in broader analysis. For the Google Scholar search query, we hand selected 4 papers directly. In total, we used 34 studies to establish the landscape, and looked further into 8 of them. Any other papers cited in the work were used for supplemental information.

IV. Survey

Research into the significance of identifiers in software engineering has been ongoing since at least 1999, which is the oldest paper in our survey [3], but has only recently gained in popularity. A histogram of the years of publication for our survey papers can be seen in figure 1. We suspect this is due to the major improvements in NLP around the time Google released its paper on the transformer in 2017 [4], Peters et al. presented ELMo in the following year [5], and finally, BERT was proposed by Devlin et al. in 2019 [6]. This aligns with the big spike in publications that same year.

We have divided the survey into three subsections, one on "traditional" algorithmic approaches, one on approaches using RNNs and LSTMs, and one on very recent advancements like transformers, and large language models.

[1] https://ieeexplore.ieee.org
[2] scholar.google.com

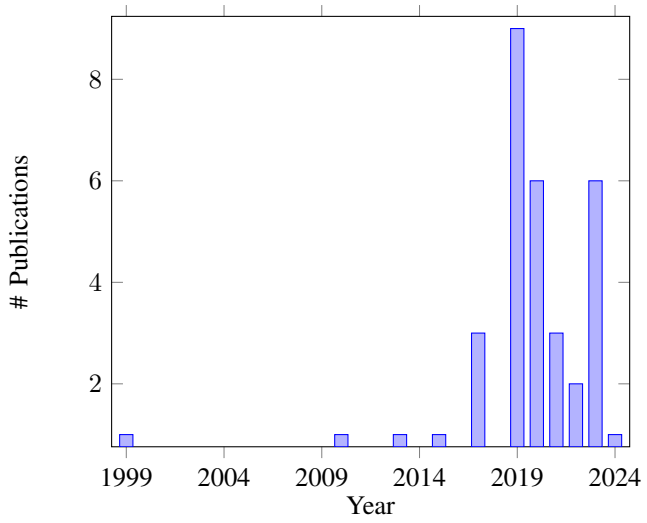

Fig. 1. Histogram of Survey Paper Publications

A. Algorithmic Approaches

The simplest approach to quantifying the correct naming of identifiers in our survey was found in the paper by Charitsis et al. [7], which focuses on the skills of computer science students. A reduced syntax backed by the Java programming language named Karel was used along with student submissions to create a simple, manually-annotated dataset of function name quality. Students submitted source code for an identical problem statement in Karel. The submitted code was compiled in-memory and instrumented, wherein the program state was compared before and after every function execution to produce a pairwise similarity score of each student function. This similarity score, along with the manually annotated quality was used to train a discriminator model producing scores on unseen identifiers. Charitsis et al. report classifier accuracies between 89.36% and 76.16%, depending on the problem hardness.

We also examined a paper which aims to tackle the out of vocabulary problem (see subsection II-B)

B. RNNs and LSTMs

***TODO*: Literature like the LSTM approach of "A Neural Model for Method Name Generation from Functional Description" by Sa Gao et al. [2]**

C. Transformers, LLMs and Beyond

***TODO*: Literature like the LLM approach of "How Well Can Masked Langauge Models Spot Identifiers That Violate Naming Guidelines?" by Johannes Villmow et al. [8]**

V. Discussion and Future Work

***TODO*: Potential for fully-integrated tools, or CI pipeline utilities. Application of cutting-edge large language models. Surprising lack of experiments with GPTs?**

VI. Conclusion

TODO

References

[1] S. Butler, M. Wermelinger, Y. Yu, and H. Sharp, "Exploring the influence of identifier names on code quality: An empirical study," in *2010 14th European Conference on Software Maintenance and Reengineering*, 2010, pp. 156–165. [Online]. Available: https://ieeexplore.ieee.org/document/5714430

[2] S. Gao, C. Chen, Z. Xing, Y. Ma, W. Song, and S.-W. Lin, "A neural model for method name generation from functional description," in *2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, Feb 2019, pp. 414–421. [Online]. Available: https://ieeexplore.ieee.org/document/8667994

[3] G. Antoniol, G. Canfora, A. Lucia, and E. Merlo, "Recovering code to documentation links in oo system," *Reverse Engineering, Working Conference on*, vol. 0, p. 136, 11 1999.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 6000–6010.

[5] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," pp. 2227–2237, jun 2018. [Online]. Available: https://aclanthology.org/N18-1202

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, Eds. Minneapolis, Minnesota: Association for Computational Linguistics, jun 2019, pp. 4171–4186. [Online]. Available: https://aclanthology.org/N19-1423

[7] C. Charitsis, C. Piech, and J. Mitchell, "Assessing function names and quantifying the relationship between identifiers and their functionality to improve them," in *Proceedings of the Eighth ACM Conference on Learning @ Scale*, ser. L@S '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 291–294. [Online]. Available: https://doi.org/10.1145/3430895.3460161

[8] J. Villmow, V. Campos, J. Petry, A. Abbad-Andaloussi, A. Ulges, and B. Weber, "How well can masked language models spot identifiers that violate naming guidelines?" in *2023 IEEE 23rd International Working Conference on Source Code Analysis and Manipulation (SCAM)*, 2023, pp. 131–142. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10356698

[9] Y. Ju, Y. Tang, J. Lan, X. Mi, and J. Zhang, "A cross-language name binding recognition and discrimination approach for identifiers," in *2023 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 2023, pp. 948–955. [Online]. Available: https://ieeexplore.ieee.org/document/10123604

[10] M. Allamanis, E. T. Barr, C. Bird, and C. Sutton, "Suggesting accurate method and class names," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, ser. ESEC/FSE 2015. New York, NY, USA: Association for Computing Machinery, 2015, p. 38–49. [Online]. Available: https://doi.org/10.1145/2786805.2786849

[11] J. Shi, Z. Yang, J. He, B. Xu, and D. Lo, "Can identifier splitting improve open-vocabulary language model of code?" in *2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 2022, pp. 1134–1138.

[12] A. Peruma, M. W. Mkaouer, M. J. Decker, and C. D. Newman, "Contextualizing rename decisions using refactorings and commit messages," in *2019 19th International Working Conference on Source Code Analysis and Manipulation (SCAM)*, 2019, pp. 74–85.

[13] C. D. Newman, A. Preuma, and R. AlSuhaibani, "Modeling the relationship between identifier name and behavior," in *2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2019, pp. 376–378.

[14] J. Zhang, J. Luo, J. Liang, L. Gong, and Z. Huang, "An accurate identifier renaming prediction and suggestion approach," *ACM Trans. Softw. Eng. Methodol.*, vol. 32, no. 6, sep 2023. [Online]. Available: https://doi.org/10.1145/3603109