JOHANNES KEPLER
UNIVERSITY LINZ

PAT'24

Measuring Identifier Quality with Machine Learning

A Survey of Existing Approaches

Marcel Simader, Melissa Frischherz

JOHANNES KEPLER
UNIVERSITY LINZ
Altenberger Straße 69
4040 Linz, Austria
jku.at

Introduction

Overview

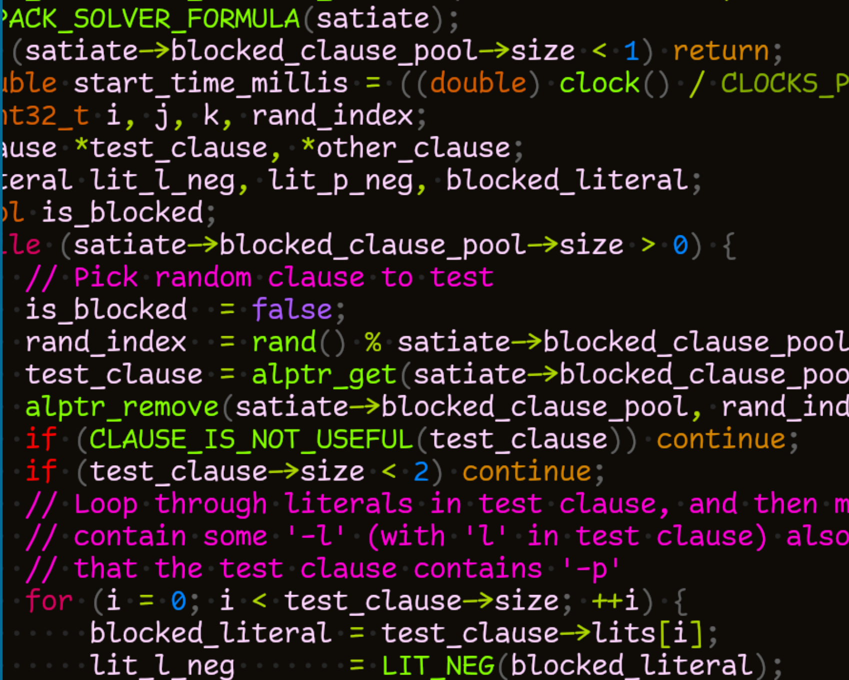
Overview

Research Question

"What Machine Learning techniques exist to (automatically) measure the quality of identifiers?"

A source code identifier is:

- a textual name for something.
- a dense unit of semantic information.
- difficult to come up with!



Overview

Research Question

"Why is identifier quality important in the software engineering context?"

- Identifiers make up ca. 70% of all source code [5]
- Poor choices for identifiers lead to:
  - less readable code
  - less maintainable code
  - overall worse quality code [3]
- Naming things is hard, because the name must *describe the meaning*

Overview

Research Question

> *"Why is identifier quality important in the software engineering context?"*

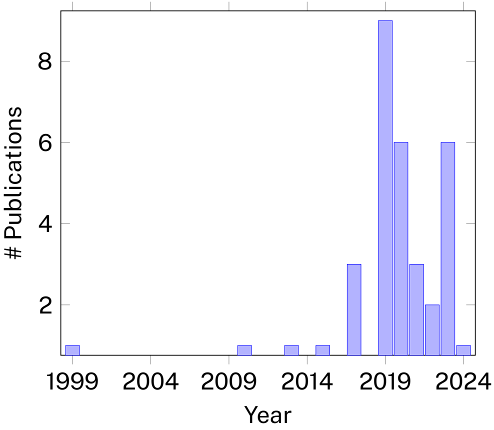
- Identifiers make up ca. 70% of all source code [5]
- Poor choices for identifiers lead to:
  - less readable code
  - less maintainable code
  - overall worse quality code [3]
- Naming things is hard, because the name must *describe the meaning*

Introduction

Methodology

Methodology

- We analyzed 34 papers in total, selected 7 for detailed analysis
- Exploration on Elicit [4], Google Scholar [6], and IEEEXplore [8]
- Search queries can be found in the paper
- We suspect the large increase around 2019 is due to Google's famous "Attention is all you need" paper



Survey

Natural Language Processing

Natural Language Processing

Problem

Extracting the information contained within an identifier is very difficult!

- Natural Language Processing (NLP) is the analysis of language as humans speak it
- Research on NLP active since the 1990s, proposed as early as the 1950s [2]

Natural Language Processing

Definition

The general goal of NLP is to find the probability of some word s_i occurring in a sentence $s_1\, s_2 \ldots s_i$:

$$P\left(s_1\, s_2 \ldots s_i\right) = P\left(s_1\right) P\left(s_2 \mid s_1\right) \cdots P\left(s_i \mid \underbrace{s_1 \ldots s_{i-1}}_{\text{“Context”}}\right) \;\; [2].$$

- We cannot compute all of these probabilities
- Context information must be condensed, somehow
- Zhang et al. use a random forest classifier with heuristic metrics to approximate these probabilities [14]

Natural Language Processing

Definition

The general goal of NLP is to find the probability of some word s_i occurring in a sentence $s_1\, s_2 \ldots s_i$:

$$P\left(s_1\, s_2 \ldots s_i\right) = P\left(s_1\right) P\left(s_2 \mid s_1\right) \cdots P\left(s_i \mid \underbrace{s_1 \ldots s_{i-1}}_{\text{“Context”}}\right) \text{ [2].}$$

- We cannot compute all of these probabilities
- Context information must be condensed, somehow
- Zhang et al. use a random forest classifier with heuristic metrics to approximate these probabilities [14]

Natural Language Processing with *n*-Grams

- The *n*-gram model "chops off" the tail of the probability distribution

Example

For $n = 2$ ("bigram"), the probability is now $P\left(s_{i-1}\, s_i\right) = P\left(s_{i-1}\right) P\left(s_i \mid s_{i-1}\right)$.

Problem

This works well for machine translation, but we cannot handle Out of Vocabulary (OOV) words... [1, 11]

- Idea: Split identifiers into tokens (e.g. get / Root / State)
  $\Longrightarrow$ fewer (new) words in training set [1, 10, 11]

Natural Language Processing with *n*-Grams

- The *n*-gram model "chops off" the tail of the probability distribution

Example

For $n = 2$ ("bigram"), the probability is now $P\left(s_{i-1}\, s_i\right) = P\left(s_{i-1}\right) P\left(s_i \mid s_{i-1}\right)$.

Problem

This works well for machine translation, but we cannot handle Out of Vocabulary (OOV) words... [1, 11]

- Idea: Split identifiers into tokens (e.g. `get` / `Root` / `State`)
  $\Longrightarrow$ fewer (new) words in training set [1, 10, 11]

Survey

Neural Networks

Neural Networks and Deep Neural Networks

Neural Networks (NNs): Based on the neuron, one input and one output layer

Deep Neural Networks (DNNs): Have at least one "hidden layer" of neurons
$\Longrightarrow$ *model of the human brain*

- NNs generally offer better accuracy for NLP applications
- We can now approximate the probability $P\left(s_1\, s_2 \ldots s_i\right)$ directly!

Neural Networks and Deep Neural Networks

- Allamanis et al. propose the Logbilinear (LBL) Language Model (LM) as solution to the *n*-grams limitations
- They show that their solution "substantially outperforms" the *n*-gram for method and type names [1]

Problem

NNs and DNNs are not well-suited for *sequential inputs*, since the input layer must scale linearly with the sequence length.

Neural Networks and Deep Neural Networks

- Allamanis et al. propose the Logbilinear (LBL) Language Model (LM) as solution to the *n*-grams limitations
- They show that their solution "substantially outperforms" the *n*-gram for method and type names [1]

Problem

NNs and DNNs are not well-suited for *sequential inputs*, since the input layer must scale linearly with the sequence length.

Recurrent Neural Networks

Recurrent Neural Networks (RNNs): Can handle a sequential input by tracking and evaluating its own state
$\Longrightarrow$ *model of human memory*

Long Short-Term Memory (LSTM): Improved kind of RNN presented by Hochreiter and Schmidhuber in 1997 [7]

- Gao et al. show that an RNN encoder-decoder model offers improved accuracy [5]
- With addition of *attention*, the model can judge the importance of input words
- With addition of *copying*, the model can use words from its input directly (another solution to the Out of Vocabulary (OOV) problem)

Recurrent Neural Networks

Recurrent Neural Networks (RNNs): Can handle a sequential input by tracking and evaluating its own state
$\Longrightarrow$ *model of human memory*

Long Short-Term Memory (LSTM): Improved kind of RNN presented by Hochreiter and Schmidhuber in 1997 [7]

- Gao et al. show that an RNN encoder-decoder model offers improved accuracy [5]
- With addition of *attention*, the model can judge the importance of input words
- With addition of *copying*, the model can use words from its input directly (another solution to the Out of Vocabulary (OOV) problem)

Transformers

Transformers: modern RNN based on series of encoder-decoder blocks with both the “multi-head” and *self-attention* mechanisms [12]
$\Longrightarrow$ *model of human memory and attention*

- Basically multiple RNNs on steroids
- Villmow et al. present a transformer Large Language Model (LLM) to check whether identifiers break naming conventions
- Their model uses 247 *million* parameters, and is trained on 6000 annotated data points [13]
- Ju et al. show that LLM can vastly outperform IntelliJ IDEA when tracking identifiers across languages [9]

Transformers

Transformers: modern RNN based on series of encoder-decoder blocks with both the “multi-head” and *self-attention* mechanisms [12]
$\Longrightarrow$ *model of human memory and attention*

- Basically multiple RNNs on steroids
- Villmow et al. present a transformer Large Language Model (LLM) to check whether identifiers break naming conventions
- Their model uses 247 *million* parameters, and is trained on 6000 annotated data points [13]
- Ju et al. show that LLM can vastly outperform IntelliJ IDEA when tracking identifiers across languages [9]

The Future

Potential Applications

Not hard to imagine:

- IDEs that can track identifiers across files and languages for intelligent refactoring
- Suggestions of variable/method/class names as you type, or the ability to get a summary of what a name means
- Improved measures of code quality for reviews (e.g. merge requests)
- Flagging of inappropriate identifiers as part of continuous integration/deployment

Open Questions

Besides the potential for larger-scale, thorough literature reviews, one big question:

Research Question

> *"Why is literature on Generative Pre-Trained Transformers (GPTs) like ChatGPT so scarce? Is the performance trade-off for LLMs worth the increased accuracy?"*

JKU

JOHANNES KEPLER
UNIVERSITY LINZ

References

References I

[1] Miltiadis Allamanis, Earl T. Barr, Christian Bird, and Charles Sutton. 2015. Suggesting accurate method and class names. In Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering (*ESEC/FSE 2015*). Association for Computing Machinery, Bergamo, Italy, pp. 38–49. ISBN: 9781450336758. DOI: 10.1145/2786805.2786849. https://doi.org/10.1145/2786805.2786849.

[2] Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, John D. Lafferty, Robert L. Mercer, and Paul S. Roossin. 1990. A Statistical Approach to Machine Translation. Computational Linguistics, 16, 2, 79–85. https://aclanthology.org/J90-2002.

References II

[3] Simon Butler, Michel Wermelinger, Yijun Yu, and Helen Sharp. 2010. Exploring the Influence of Identifier Names on Code Quality: An Empirical Study. In 2010 14th European Conference on Software Maintenance and Reengineering, pp. 156–165. DOI: 10.1109/CSMR.2010.27. https://ieeexplore.ieee.org/document/5714430.

[4] Elicit. 2023. Elicit: The AI Research Assistant. (January 24, 2023). https://elicit.com.

References III

[5] Sa Gao, Chunyang Chen, Zhenchang Xing, Yukun Ma, Wen Song, and Shang-Wei Lin. 2019. A Neural Model for Method Name Generation from Functional Description. In 2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER). (February 2019), pp. 414–421. DOI: 10.1109/SANER.2019.8667994. https://ieeexplore.ieee.org/document/8667994.

[6] Google. 2024. Google Scholar. (2024). https://scholar.google.com.

[7] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-term Memory. Neural computation, 9, (December 1997), 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

References IV

[8] IEEE and IET. 2024. IEEEXplore. (2024). https://ieeexplore.ieee.org.

[9] Yue Ju, Yixuan Tang, Jinpeng Lan, Xiangbo Mi, and Jingxuan Zhang. 2023. A Cross-Language Name Binding Recognition and Discrimination Approach for Identifiers. In 2023 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), pp. 948–955. DOI: 10.1109/SANER56733.2023.00115. https://ieeexplore.ieee.org/document/10123604.

References V

[10] Rafael-Michael Karampatsis, Hlib Babii, Romain Robbes, Charles Sutton, and Andrea Janes. 2020. Big code != big vocabulary: open-vocabulary models for source code. In Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering (*ICSE '20*). Association for Computing Machinery, Seoul, South Korea, pp. 1073–1085. ISBN: 9781450371216. DOI: 10.1145/3377811.3380342. https://doi.org/10.1145/3377811.3380342.

[11] Jieke Shi, Zhou Yang, Junda He, Bowen Xu, and David Lo. 2022. Can Identifier Splitting Improve Open-Vocabulary Language Model of Code? In 2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), pp. 1134–1138. DOI: 10.1109/SANER53432.2022.00130.

References VI

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Proceedings of the 31st International Conference on Neural Information Processing Systems (*NIPS'17*). Curran Associates Inc., Long Beach, California, USA, pp. 6000–6010. ISBN: 9781510860964.

References VII

[13] Johannes Villmow, Viola Campos, Jean Petry, Amine Abbad-Andaloussi, Adrian Ulges, and Barbara Weber. 2023. 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), pp. 131–142. DOI: 10.1109/SCAM59687.2023.00023. https://ieeexplore.ieee.org/abstract/document/10356698.

References VIII

[14] Jingxuan Zhang, Junpeng Luo, Jiahui Liang, Lina Gong, and Zhiqiu Huang. 2023. An Accurate Identifier Renaming Prediction and Suggestion Approach. ACM Trans. Softw. Eng. Methodol. 32, 6, Article 148, (September 2023), 51 pages. ISSN: 1049-331X. DOI: 10.1145/3603109. https://doi.org/10.1145/3603109.