Proceedings of the 4th International Conference on Signal Processing and Machine Learning
DOI: 10.54254/2755-2721/50/20241673

Review of method name recommendation

Hui Zhang[1,2], Zhenjie Luo[1]

[1]School of Cybersecurity Northwestern Polytechnical University, Xi'an, China

[2]Correspondence: Hui Zhang, zhanghui9903@mail.nwpu.edu.cn

Abstract. In software engineering projects, method name recommendation is a critical task aimed at assisting developers in selecting appropriate method names to enhance code readability and maintainability. Given the significance of method name recommendation, numerous researchers have delved into improving the quality of method names. In this paper, we systematically review the existing method name recommendation techniques and conduct a comprehensive analysis of their principles, characteristics, and performance. Furthermore, we summarize the comparisons and evaluations among various methods to aid researchers in choosing method name recommendation techniques that suit their specific requirements. Lastly, we highlight the future research directions in this field. Through this review, readers will gain a clearer understanding of the current state of method name recommendation, providing valuable guidance for further research and development.

Keywords: Method Name Recommendation, Machine Learning, Information Retrieval

1. Introduction

With the flourishing development of the computer software domain, software development has transformed from an isolated activity into a highly collaborative practice. Developers must work together to build increasingly vast and complex software systems. In this environment, the selection of method names becomes critically important. The readability of method names plays a vital role in helping developers understand the functionality of the program and the interactions between different modules. Method names provide the most intuitive information for developers to comprehend and maintain a program, making appropriate method names crucial for the comprehensibility of software code.

Concise and meaningful method names not only make the code more understandable but also contribute to reducing error rates, enhancing development efficiency, and promoting collaboration. Accurate, clear, and semantically consistent method names aid developers in quickly grasping the function and purpose of methods, reducing the likelihood of confusion and misunderstanding. They also facilitate the rapid integration of new team members, lowering the learning curve and increasing the overall productivity of the team.

Method name recommendation is not merely a part of the code but rather a fundamental element in the software development lifecycle, laying a solid foundation for building high-quality, maintainable software systems. Despite the significant importance of good method names in software development and maintenance, the naming quality of software methods in practical applications is often subpar. This

DOI: 10.54254/2755-2721/50/20241673

phenomenon is prevalent in many projects, resulting in code that is difficult to understand, challenging to maintain, lower development efficiency, and teamwork issues.

In current software development practices, the choice of method names often relies on developers' subjective experience and personal style, lacking clear guiding principles. This lack of standardization makes it difficult to ensure the consistency and quality of method names. Furthermore, software projects typically involve the collaboration of multiple developers, further intensifying the challenge of maintaining consistency in method names. How to standardize and improve the quality of method names has become an urgent issue to address.

This paper represents the first comprehensive review on the subject of method name recommendation. This review not only provides a clear depiction of the current state of method name recommendation but also outlines potential avenues for future research in this continually evolving field. The purpose of this paper is to offer a comprehensive perspective for relevant practitioners and researchers, aiding them in better understanding and addressing the challenges of method name quality.

The structure of this paper is as follows: In Chapter 2, we introduce the definition and background overview of method name recommendation. In Chapter 3, we present research advancements in method name recommendation categorized into two distinct technological approaches. Chapter 4 discusses the challenges faced in this field and offers directions for future development. Chapter 5 summarizes the work presented in this paper.

2. Basic Concepts

Method names are one of the key elements in software code and serve multiple roles. Firstly, method names act as the entry points in code, providing an interface to pass control flow to the respective method, guiding the execution flow of the code. Additionally, method names serve as self-descriptive elements of the code, ideally reflecting the function and purpose of the method clearly, thereby offering other developers an initial understanding of the method. Method names are also used for code documentation, providing insights to other developers on how the method is designed to work, its inputs and outputs, and how to use it correctly.

The principles of method name recommendation involve providing suggestions for method names by analyzing code and context information. Its objective is to automatically or semi-automatically offer meaningful, clear, and consistent method names to enhance code quality and maintainability. The technologies involved in method name recommendation include text mining, natural language processing, machine learning, and more, all of which analyze code and context information to provide recommendations for method names. Text mining techniques extract keywords, phrases, and context information from the code to offer meaningful suggestions for method names. Natural language processing techniques analyze code and context information to generate meaningful, clear, and consistent method names. Machine learning techniques learn from method names in code repositories to provide recommendations for new methods. The combination of these technologies can improve the quality and readability of method names, making the code more understandable and maintainable.

When selecting an appropriate method name, several key attributes need to be considered to ensure its effectiveness and contribution. As shown in Figure 1, These attributes include:

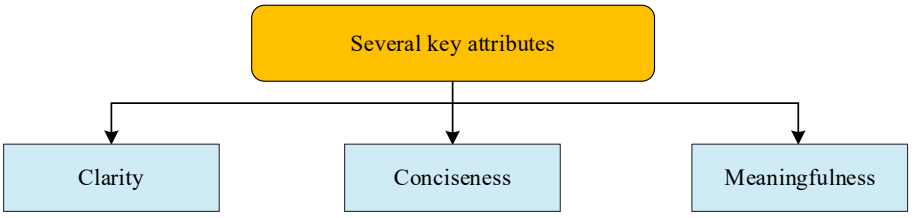

Figure 1. Several key attributes

DOI: 10.54254/2755-2721/50/20241673

Clarity: A clear method name should effectively convey the function and purpose of the method, making it easily understandable. Clarity requires the method name to provide an accurate description of the method's operation so that other developers can quickly grasp its purpose.

Conciseness: A concise method name is short and to the point, without being overly long or complex. Conciseness is crucial for improving code readability because shorter names are easier to scan and understand. Lengthy or convoluted method names can make the code cluttered and increase the time cost of understanding the code. Concise method names contribute to code clarity and maintainability.

Meaningfulness: Method names must have practical significance, accurately reflecting the function and purpose of the method. Meaningful method names help reduce the likelihood of misunderstanding, as other developers can understand the purpose of the method through its name without delving deeply into the code. Meaningful method names contribute to code quality and collaboration by providing clearer insights into the code.

In software development, the choice of method names is a critical decision that directly impacts code quality and maintainability. Therefore, developers should carefully weigh these attributes to ensure the effectiveness and practical significance of method names.

3. Method Name Recommendation Techniques

In this chapter, we will delve into various technical methods for method name recommendation, As shown in Figure 2, including information retrieval-based methods, machine learning methods, and other approaches. These techniques play a crucial role in providing suggestions and recommendations for method names, assisting developers in choosing more suitable names.

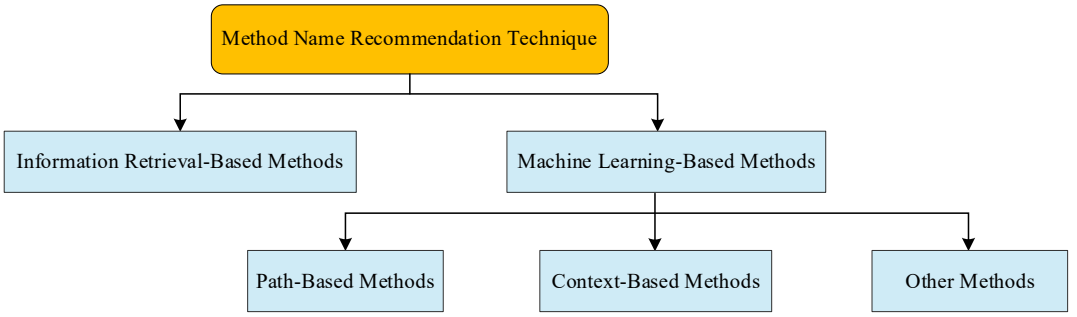

Figure 2. Method Name Recommendation Techniques

3.1. Information Retrieval-Based Methods

Information retrieval-based methods rely on text matching and similarity scores to recommend method names, typically using techniques like TF-IDF to assess the relevance between texts. Essentially, these methods are based on the theory of code similarity.

Initially, Deissenboeck et al. [1] created a function name recommendation model based on the mapping relationship between function naming rules and recommended names. This model can be used to generate clear and accurate function names. Additionally, they constructed a corresponding name dictionary within the model to ensure that various identifier names in the program remain consistent throughout the software's lifecycle.

Jiang et al. [2] proposed a function name recommendation method based on code search, HeMa, which retrieves function names with similar return types and parameters in a constructed code database and derives corresponding function names based on this. The main challenge of the HeMa method is that it cannot recommend any method names when no method with the same return type and input parameters is found in the reference dataset.

Gao et al. [3] introduced a function name recommendation method based on code library and feature matching. This method relies on creating a function library from open-source software, retrieving functions from the library that are similar to the one being recommended, and performing a secondary

DOI: 10.54254/2755-2721/50/20241673

parsing of the retrieval results to obtain annotated terms. These annotated terms are then associated with the feature code of the recommended function, automatically suggesting suitable function names.

Liu et al. [4] proposed a function name consistency checking method based on similarity calculation. This method utilizes paragraph vectors and convolutional neural networks to extract deep representations of method names and bodies separately. For a given function m, they retrieve a set of function names NS that are similar to m in the name vector space and a set of function names BS that are similar to m in the entity vector space. However, in many cases, two methods with similar entities are given different names because they may have different semantics or are used for various tasks.

Yang et al. [5] applied the principles of information retrieval to introduce a pattern-based method name recommendation method called NamPat. NamPat initially retrieves the most similar methods from the training data by estimating the similarity of the target method's body code. It then uses the name of the most similar method as a pattern guide and combines it with the context information of the target method to execute method name recommendation.

The key advantages of these information retrieval-based methods are their simplicity, ease of use, and low computational costs. However, they also have notable disadvantages. The manual effort required to construct a function retrieval database can be high, and retrieval efficiency may decrease as the database size grows. Another significant drawback of information retrieval-based methods is that they cannot generate new names that are not present in the training data since they rely on searching within the existing name set.

3.2. Machine Learning-Based Methods

Machine learning-based methods can overcome key limitations of information retrieval-based methods as they have the ability to generate new names. Machine learning methods use algorithms and models to learn from existing method name data and predict recommended method names based on input context and features. This involves the use of natural language processing techniques, such as word embeddings and recurrent neural networks (RNNs), to capture semantic information and context. While machine learning methods typically require a substantial amount of training data, they can provide more accurate and semantic method name suggestions as they can understand more complex semantics and grammar rules.

3.2.1. Path-Based Methods

Alon et al. [6] introduced a universal path-based function representation method called Path-Rep. The primary idea behind Path-Rep is to use paths within abstract syntax trees (AST) to represent programs. This allows the model to effectively utilize the structured nature of the source code rather than treating it as a simple sequence of tokens. Code2vec [7] and Code2seq [8], also proposed by Alon and collaborators, represent method bodies as sets of combined AST paths and use these path representations to predict method names.

In their work [8], Alon and others considered the unique syntactic structure of source code. They sampled paths within the abstract syntax tree of code fragments, encoded these AST paths using bidirectional LSTM, then averaged the representations of all paths to obtain the final representation of the program encoder. Another LSTM was used as a decoder to generate the output, which could be a method name or code summary.

Zügner et al. [9] introduced a Transformer-based, language-agnostic code representation model called Code Transformer. Peng et al. [10] also proposed a Transformer-based code representation learning model. This model integrated two types of paths (absolute paths and relative paths) into a unified Transformer framework and investigated their interactions. Both Zügner et al. [9] and Peng et al. [10] considered AST paths extracted from method bodies as context and used path-based representations for predicting method names.

These path-based representations and models allow for a more structured and semantic understanding of code, which can lead to better method name recommendations and code summarization.

DOI: 10.54254/2755-2721/50/20241673

3.2.2. Context-Based Methods

Allamanis and colleagues introduced a logarithmic bilinear neural language model [11]. This model maps each identifier within a given method body to high-dimensional vectors in continuous space to learn which names are semantically similar. It recommends function names that are most aligned with the functionality of the function. Later, Allamanis et al. [12] treated the method name recommendation problem as an extreme summarization task from method bodies to short text. To generate code summaries, they proposed a convolutional attention neural network for feature extraction from the context. This network performs convolutions on input tokens within the method body, allowing the model to learn high-level functional features of the source code to recommend method names.

Parsa and colleagues [13] pointed out that Code2vec cannot capture semantic information in method names, and Code2seq cannot handle long method names. Their method, proposed in the paper, addresses these issues by using source code metrics and naming patterns, thus improving the accuracy of method name recommendation.

Nguyen and colleagues [14] found that most keywords present in method names can be located in three types of contextual information: the body of the given function, the interface (function parameter types and return types), and the enclosing class. They proposed a method name consistency checking and recommendation tool called MNire based on a Seq2Seq model. This tool treats the method name generation task as summarizing the program entity names within the method body and the enclosing class name, using program entity tokens from three contexts (i.e., implementation context, interface context, and encapsulation context) to generate method names. Building on MNire, Li et al. [15] considered using more context in method name recommendations and introduced the DeepName tool. Wang et al. [16] identified limitations in MNire when dealing with less content-rich methods, so they developed Cognac, which incorporates caller/callee context as prior knowledge for completing method name recommendations.

Inspired by DeepName and Cognac, Liu et al. [17] proposed a method name recommendation method called GTNM based on a global transformer. GTNM treats the method name recommendation task as abstract text summarization, where tokens from different levels of context are considered as input, and subtokens in the method name are regarded as the target summary from the input sequence. It employs attention mechanisms to allow the model to focus on different contexts during the decoding process.

3.2.3. Other Methods

Wang et al. [18] introduced a deep neural network called Liger. Their approach blends symbolic information with specific execution paths to capture program semantics.

Yonai and colleagues developed the method name recommendation tool Mercem [19], which is based on call graph embedding. Mercem uses graph embeddings on Aggregate Call Graphs (ACG) to represent method bodies. The idea is to recommend method names to software developers by utilizing method embeddings obtained from caller-callee relationships.

Wang et al. [20] introduced a novel graph neural architecture called GINN for learning distributed representations of programs. The authors applied GINN to the task of method name recommendation and achieved promising results.

These methods leverage graph-based representations, combining symbolic and concrete information to enhance the recommendation of method names, and have shown effectiveness in various software development contexts.

4. Challenges and Prospects

4.1. Challenges

Diversity and Consistency: One of the challenges is to achieve both diversity and consistency in method name recommendations. Diversity requires the system to generate different types of method names, while consistency demands that the names align with the code's functionality.

DOI: 10.54254/2755-2721/50/20241673

Multi-Language Support: Method name recommendation needs to consider various programming languages. Each language has its unique naming conventions and rules, making it a challenge to provide effective method name recommendations across multiple languages.

Data Sparsity: Data is crucial for the success of deep learning methods, but many codebases in projects are small or limited. The challenge is to train effective method name recommendation models in situations with sparse data.

4.2. Prospects

Multi-Modal Method Name Recommendation: Future research can explore the integration of text and code in multi-modal method name recommendation. This will enable models to better understand the code context and generate more semantically meaningful method names.

Data Augmentation and Transfer Learning: Studies can investigate data augmentation techniques to expand datasets and utilize transfer learning methods to leverage knowledge learned from one project to improve performance in other projects.

Tool Integration: Future research can integrate method name recommendation into existing development tools to assist developers in writing high-quality code more effortlessly.

5. Conclusion

This paper is the first comprehensive review of method name recommendation, systematically introducing the challenges, methods, and future directions in this field. Our review provides researchers with a comprehensive perspective to help them better understand the current state, challenges, and future trends in this area. We hope that this review will inspire more innovation and research, providing further support for code writing and maintenance.

References

[1] Gethers M, Savage T, Di Penta M, et al. CodeTopics: which topic am I coding now?[C]//Proceedings of the 33rd International Conference on Software Engineering. 2011: 1034-1036.

[2] Jiang L, Liu H, Jiang H. Machine learning based recommendation of method names: How far are we[C]//2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, 2019: 602-614.

[3] Gao Yuan, Liu Hui, Fan Xiaozhong, et al. Function name recommendation method based on code base and feature matching [J]. Journal of Software, 2015, 26(12): 3062-3074.

[4] Liu K, Kim D, Bissyandé T F, et al. Learning to spot and refactor inconsistent method names[C]//2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE). IEEE, 2019: 1-12.

[5] Yang Y, Xu L, Yan M, et al. A Naming Pattern Based Approach for Method Name Recommendation[C]//2022 IEEE 33rd International Symposium on Software Reliability Engineering (ISSRE). IEEE, 2022: 344-354.

[6] Alon U, Zilberstein M, Levy O, et al. A general path-based representation for predicting program properties[J]. ACM SIGPLAN Notices, 2018, 53(4): 404-419.

[7] Alon U, Zilberstein M, Levy O, et al. code2vec: Learning distributed representations of code[J]. Proceedings of the ACM on Programming Languages, 2019, 3(POPL): 1-29.

[8] Alon U, Brody S, Levy O, et al. code2seq: Generating sequences from structured representations of code[J]. arXiv preprint arXiv:1808.01400, 2018.

[9] Zügner D, Kirschstein T, Catasta M, et al. Language-agnostic representation learning of source code from structure and context[J]. ar**v preprint ar**v:2103.11318, 2021.

[10] Peng H, Li G, Wang W, et al. Integrating tree path in transformer for code representation[J]. Advances in Neural Information Processing Systems, 2021, 34: 9343-9354.

DOI: 10.54254/2755-2721/50/20241673

[11] Allamanis M, Barr E T, Bird C, et al. Suggesting accurate method and class names[C]//Proceedings of the 2015 10th joint meeting on foundations of software engineering. 2015: 38-49.
[12] Allamanis M, Peng H, Sutton C. A convolutional attention network for extreme summarization of source code[C]//International conference on machine learning. PMLR, 2016: 2091-2100.
[13] Parsa S, Zakeri-Nasrabadi M, Ekhtiarzadeh M, et al. Method name recommendation based on source code metrics[J]. Journal of Computer Languages, 2023, 74: 101177.
[14] Nguyen S, Phan H, Le T, et al. Suggesting natural method names to check name consistencies[C]//Proceedings of the acm/ieee 42nd international conference on software engineering. 2020: 1372-1384.
[15] Li Y, Wang S, Nguyen T. A context-based automated approach for method name consistency checking and suggestion[C]//2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE). IEEE, 2021: 574-586.
[16] Wang S, Wen M, Lin B, et al. Lightweight global and local contexts guided method name recommendation with prior knowledge[C]//Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 2021: 741-753.
[17] Liu F, Li G, Fu Z, et al. Learning to recommend method names with global context[C]//Proceedings of the 44th International Conference on Software Engineering. 2022: 1294-1306.
[18] Wang K, Su Z. Blended, precise semantic program embeddings[C]//Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation. 2020: 121-134.
[19] Yonai H, Hayase Y, Kitagawa H. Mercem: Method name recommendation based on call graph embedding[C]//2019 26th Asia-Pacific Software Engineering Conference (APSEC). IEEE, 2019: 134-141.
[20] Wang Y, Wang K, Gao F, et al. Learning semantic program embeddings with graph interval neural network[J]. Proceedings of the ACM on Programming Languages, 2020, 4(OOPSLA): 1-27.