CodeMatcher: Searching Code Based on Sequential Semantics of Important Query Words

An Extended Abstract of a Paper Published in ACM Transactions on Software Engineering and Methodology

Chao Liu
School of Big Data & Software Engineering, Chongqing University
Chongqing, China
liu.chao@cqu.edu.cn

Xin Xia
Software Engineering Application Technology Lab, Huawei
Hangzhou, China
xin.xia@acm.org

David Lo
School of Information Systems, Singapore Management University
Singapore
davidlo@smu.edu.sg

Zhiwei Liu
Baidu Inc.
Shanghai, China.
zhiweiliu03@baidu.com

Ahmed E. Hassan
Software Analysis and Intelligence Lab, Queen's University
Ontario, Canada
ahmed@queensu.ca

Shanping Li
College of Computer Science and Technology, Zhejiang University
Hangzhou, China
shan@zju.edu.cn

ABSTRACT

To accelerate software development, developers frequently search and reuse existing code snippets from a large-scale codebase, e.g., GitHub. Over the years, researchers proposed many information retrieval (IR) based models for code search, but they fail to connect the semantic gap between query and code. An early successful deep learning (DL) based model DeepCS solved this issue by learning the relationship between pairs of code methods and corresponding natural language descriptions. Two major advantages of DeepCS are the capability of understanding irrelevant/noisy keywords and capturing sequential relationships between words in query and code.

In this paper, we proposed an IR-based model CodeMatcher [3] that maintains the advantageous features in DeepCS as described above. The IR technique is adopted because it can avoid time-consuming training and improve code search efficiency, and researchers do not need to take a lot of efforts for manually tuning complex parameters in DL-based model. Generally, CodeMatcher leverages Elasticsearch, a Lucene-based text search engine to index codebase and perform fuzzy search with the identified important keywords from search queries. To improve query understanding, CodeMatcher removes noisy keywords according to some collected metadata and replaces irrelevant keywords with appropriate synonyms extracted from the codebase. To optimize the ranking of code methods searched by Elasticsearch, we designed two strategies (S_{name} and S_{body}) to rerank the code methods. For a pair of query and method, S_{name} measures the semantic similarity between the query and method name. Meanwhile, S_{body} measures the semantic similarity between the query and method body (i.e., the method implementation part). The similarity measurements consider the semantically related important words between queries and code methods and their sequential relationships.

Our study investigated the following research questions (RQs):

RQ1: *Can CodeMatcher outperform the baseline models?* We tested CodeMatcher and the models DeepCS [2], CodeHow [4], and UNIF [1] on a large-scale testing data with ~41k repositories. Experimental results showed that CodeMatcher achieves an MRR of 0.60, substantially outperforming DeepCS, CodeHow, and UNIF by 82%, 62%, and 46% respectively.

RQ2: *Is CodeMatcher faster than the baseline* models? CodeMatcher needs no model training like DeepCS and UNIF, and it only takes 0.3s for code search per query, which is over 1.2k and 1.5k times faster than two DL-based models DeepCS and UNIF respectively. Besides, CodeMatcher works 8 times faster than the IR-based model CodeHow.

RQ3: *How do the CodeMatcher components contribute to the code search performance?* The CodeMatcher consists of three components, namely query understanding, fuzzy search, and reranking. The query understanding mainly collects metadata for the other two components. When removing the reranking component, the search performance is reduced by 22% in terms of MRR (from 0.60 to 0.47). However, the model with only fuzzy search component still outperforms baseline models DeepCS, CodeHow, and UNIF by 42%, 27%, and 15% respectively. Therefore, all the components are necessarily required for CodeMatcher.

RQ4: *Can CodeMatcher outperform existing online code search engines?* The proposed model CodeMatcher outperforms the popular online code search engines, GitHub search and Google search, by 46% and 33% in terms of MRR respectively. Moreover, we also found that combining Google search to CodeMatcher can further improve the MRR of CodeMatcher from 0.60 to 0.64. These results imply the merit of CodeMatcher in practical usage.

The main contributions of this study are:

- We propose an IR-based model CodeMatcher that not also runs fast but also inherits the advantages of the DL-based model DeepCS. It can better understand query semantics by addressing irrelevant/noisy words, and correctly map the

ICSE '22, May 21-29, 2022, Pittsburgh, PA, USA
© 2022 Copyright held by the owner/author(s).
ACM ISBN xxxxxx.
https://doi.org/10.1109/ICSE.2021.2918536

query-code semantics by capturing the sequential relationship between important words.

- We evaluate the effectiveness of CodeMatcher on a large-scale codebase collected by us from GitHub. The experimental results show that CodeMatcher substantially outperforms three existing models (DeepCS, CodeHow, and UNIF).
- We compare CodeMatcher with two existing online search engines, GitHub and Google search. CodeMatcher also shows substantial advantages over these two popular online search engines in code search.

The full paper is published in ACM Transactions on Software Engineering and Methodology, and can be found at: https://dl.acm.org/doi/abs/10.1145/3465403

Please cite the following paper: Chao Liu, Xin Xia, David Lo, Zhiwe Liu, Ahmed E Hassan, and Shanping Li. CodeMatcher: Searching Code Based on Sequential Semantics of Important Query Words. ACM Transactions on Software Engineering and Methodology (TOSEM). 2021, 31(1):1-37.

KEYWORDS

Code Search, Code Indexing, Mining Software Repositories, Information Retrieval

REFERENCES

[1] Jose Cambronero, Hongyu Li, Seohyun Kim, Koushik Sen, and Satish Chandra. 2019. When deep learning met code search. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 964–974.

[2] Xiaodong Gu, Hongyu Zhang, and Sunghun Kim. 2018. Deep code search. In *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*. IEEE, 933–944.

[3] Chao Liu, Xin Xia, David Lo, Zhiwe Liu, Ahmed E Hassan, and Shanping Li. 2021. CodeMatcher: Searching Code Based on Sequential Semantics of Important Query Words. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 31, 1 (2021), 1–37.

[4] Fei Lv, Hongyu Zhang, Jian-guang Lou, Shaowei Wang, Dongmei Zhang, and Jianjun Zhao. 2015. Codehow: Effective code search based on api understanding and extended boolean model (e). In *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 260–270.