A Semantic Code Search Method Based on Program Conversion

Yan Tao[1], Le Wei[1,2], Hongping Shu[2]

[1] *School of Software Engineering, Chengdu University of Information Technology, Chengdu, Sichuan 610225, China*
[2] *Automatic Software Generation & Intelligence Service Key Laboratory of Sichuan Province, Chengdu 610225, China*

Abstract

Aiming at the problem that code fragments can't be captured accurately and quickly due to ignoring semantic information and structural information of source code in code search task, A semantic code search method based on program conversion is proposed (Semantic Code Search based on Program Conversion, SCSPC). The SCSPC diversified the data of the acquired code fragments through data enhancement, changed the program through variable renaming, exchanging two independent statements, circular exchange, inserting exception capture and if equivalent replacing switch statement, and trained the CodeBERT model with mixed objective functions (masking language modeling and replacing token detection) to generate sentence vectors with rich semantics for the code fragments and natural languages, and compared the similarity of the vectors to complete the code search. Experimental results show that compared with SWIM, QECK and CODEnn models, the average reciprocal ranking and hit rate (S@10) of SCSPC method are increased by 2% and 0.041 respectively.

Keywords

Code Search, CodeBERT, Programming Transformation, Semantic Code

1. Introduction

In modern software development process, software reuse has been applied to various fields. The same function may exist in different domains or different software systems, and the code to implement the function is also similar. Developers usually search for code in a large number of procedural code libraries, so as to save development time and efficiency.

At present, code search is mainly based on information retrieval methods and deep learning methods. Methods based on information retrieval focus on how to generate correct keywords and calculate the matching degree between them. For example, CodeHow proposed by Lv et al. [1], this model is based on the natural language similarity and the influence of API on code search, and understands the query by identifying the API that the query may refer to. After determining the potential API of the query, the information of the API is merged into the process of code search. Lu et al. [2] proposed a synonym expansion query model generated by WordNet [3]. Iman et al. [4] proposed a pattern-based search technology, which uses the clone detection method based on vector space model to support the discovery of working code instances and search out all types of statements (such as control flow, data flow and API).

Current code search methods still have problems. First, they ignore that codes may come from different dimensions, which makes it difficult to cover all perspectives with a single code representation. Second, it is difficult for a single natural language query to express the intentions of different users, which makes the search results inaccurate. Aiming at the above problems, this paper proposes a code search method based on program transformation. The method uses program transformation to enhance data and diversify code fragments. The CodeBERT model is used to map natural language and code

AHPCAI2022@2nd International Conference on Algorithms, High Performance Computing and Artificial Intelligence
EMAIL: Corresponding author's e-mail: 897123966@qq.com (Yan Tao)

snippets into the same vector space, and the similarity between the vectors is calculated and sorted to search for the corresponding code snippet for the user.

2. The model of CodeBERT

Pretrained models are essentially a method of transfer learning. The most typical pre-trained model is BERT [5] (Bidirectional Encoder Representation from Transformers). BERT uses a bidirectional encoder from Transformer [6], which is geared towards textual languages but is not suitable for representing semantic relationships between bimodal languages.

The CodeBERT [7] model captures the semantic association between natural language and code snippet, and can quickly complete tasks such as semantic similarity search. The CodeBERT model is based on a Transformer network with 12 layers. Its pre-training completes two tasks: Masked Language Model (MLM) and Substitution Token Detection (RTD) task. The masked language model is to randomly delete a word in a sentence and then judge what the deleted word is, a task pretrained using NL-PL pairs. As shown in Figure 1, in the pre-training phase, [CLS] is placed at the beginning of the natural language sentence to do binary classification. Separate natural language and code snippets with flags. The input is of the form {[CLS], NL, [SEP], PL, [EOS]}. Consider natural language as a sequence of words and code snippets as a whole set of tokens. The output is the Embedding value and the [CLS] value for each Token. The Token detection task uses two generators combined with the context to generate the token at [MASK], and trains the discriminator to predict whether the token in the sentence is replaced or not. This task is pretrained using single-modal data, and the training process is shown in Figure 2.

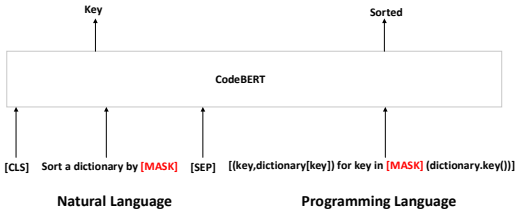

Figure 1. The model of masked language.

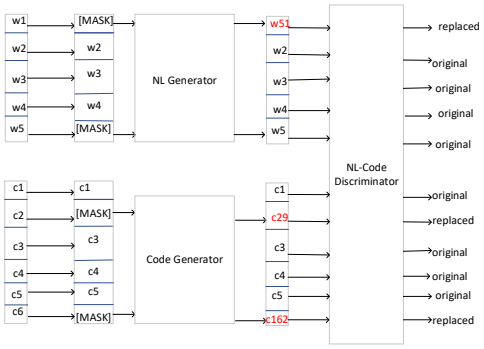

Figure 2. The tasks of token detection.

3. Code search method based on program transformation

3.1. The framework of the method

The ultimate goal of method framework code search is to find code snippets that match the user's expectations. Program transformation based Semantic code search (SCSPC) captures the semantic relationships between natural languages and programming languages and generates a common representation. The work of this paper mainly includes semantic-preserving code program transformation, model training, and code search result matching. Its overview diagram is shown in Figure 3. Firstly, the SCSPC method uses data augmentation to transform the source code. The specific steps are to rename the source code variables, swap the position of independent statements, loop exchange, add capture statements, and equivalently replace switch statements with if. Each transformation has different effects on the structure of the method, so as to improve the diversity of the training data. Then, natural language and code snippets are input into the CodeBERT model, which are mapped into sentence vectors and code block vectors respectively. Finally, the cosine distance between the code block vector and the sentence vector is calculated by the cosine similarity, and the code search is completed by sorting the similarity.

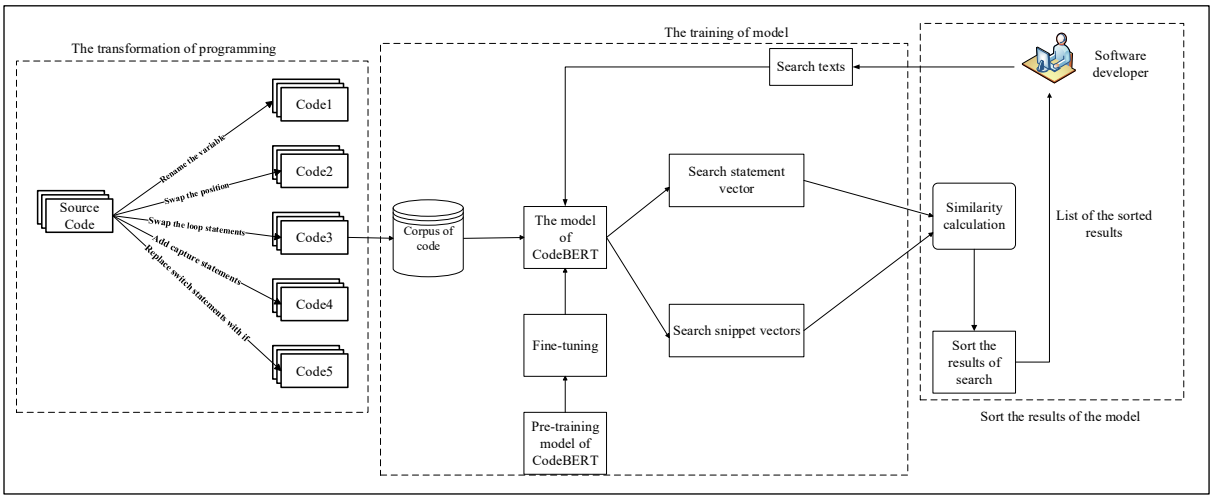

Figure 3. Overview diagram of semantic code search methods based on program transformations.

3.2. The model training of the method

Model training focuses on capturing the semantic connections between queries and code snippets and generating a common representation, as shown in Figure 4.

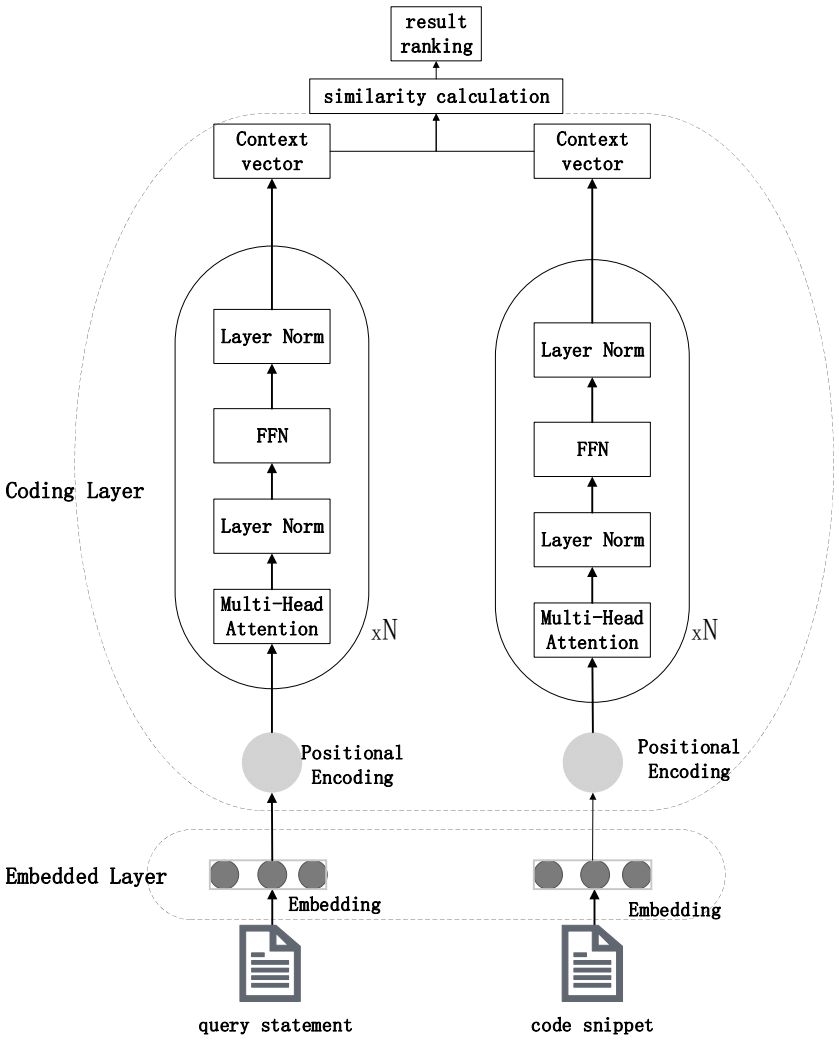

Figure 4. The process of model training.

The model mainly consists of embedding layer and encoding layer. Within the embedding layer, input queries and code snippets are mapped into a high-dimensional vector space. Set the maximum length of the input query to 1024. Think of it as a sequence of words, where each query uses <CLS> and <EOS> to represent the beginning and end of the query. Use <SEP> to split words. For a code fragment, think of it as a sequence of tokens. The encoding layer learns code snippets c and natural language d using a bidirectional encoder. The main calculation formula is shown in (1). PE is the position embedding and i denotes the dimension of the embedding. Linear stands for linear layer, Q, K, V are three vectors in the attention mechanism, which are query vector, key vector, and value vector. LayerNorm represents the normalization layer, RELU is the activation function, and using RELU can speed up the convergence of the network.

$$\begin{cases} PE_{(\mathrm{pos},2i)}=\sin(pos/10000^{2i/d}) \\ PE_{(pos,2i+1)}=\cos(pos/10000^{2i/d}) \\ X = Embedding + PositionalEncoding \\ Q = Linear(X) = XW_Q \\ K = Linear(X) = XW_k \\ V = Linear(X) = XW_V \\ X_{attention} = X_{attention} + X \\ X_{attention} = LayerNorm(X_{attention}) \\ X_{hidden} = Linear(RULU(Linear(X_{attention}))) \end{cases} \tag{1}$$

4. Experiment

4.1. Data of the experiment

In order to ensure that the collected source code has real reliability. The source code of this paper is from the most popular hosting open source website platform Github and the Deep Program group of Microsoft Research-Cambridge jointly launched the dataset CodeSearchNet for code representation learning [8]. The CodeSearchNet dataset is composed of 2 million annotation-code pairs from open source libraries. Specifically, comments are top-level function or method comments, and code is the entire function or method. In this experiment, the Java language from the CodeSearchNet Challenge was selected as the dataset. It contains 500,754 pairs of functional Java code snippets and their descriptions. 450, 439 pairs were used as training, 33, 658 pairs were used as validation set, and 28, 910 pairs were used as test set.

4.2. Evaluation criteria of the experiment

In this paper, SuccessRate@k and MRR (Mean Reciprocal Rank) are used to verify the effectiveness of the SCSPC method. The formula for calculating the average reciprocal rank is given in (2). Q denotes the set of automatically evaluated queries, and ranki is the rank corresponding to the ith query. The higher the value of MRR, the better the performance of the code search.

$$\mathrm{MRR} = \frac{1}{Q}\sum_{i=1}^{|Q|}\frac{1}{rank_i} \tag{2}$$

SuccessRate@k denotes the percentage of queries for which more than one correct fragment successfully exists among the top k code fragments returned by the search model, and is calculated as shown in (3). Where Q is the query set in the automatic evaluation, Rankq is the highest Rank of the hit code snippet in the search results, and σ is a function that returns 1 if the rank value of the q query is less than k, and 0 otherwise. This evaluation metric is very important. Because a good search engine should be able to find the code snippets that developers need by searching fewer search results. In this

experiment, the results were evaluated for k of 1, 5, and 10. If the value is higher, the model performance of code search is better.

$$SuccessRate@k = \frac{1}{|Q|}\sum_{q=1}^{Q}\sigma\left(rank_q \le k\right) \tag{3}$$

4.3. Results of the experiment

In order to verify the effectiveness of the SCSPC method, SWIM [9], QECK [10] and CODEnn are selected as the three benchmark methods to compare this experiment, and the code repository in Section 4.1 is used to construct the training set, validation set and test set.

In this experiment, the value of search result K is set as 1, 5, and 10. A value of 1 represents the probability that the correct search result appears in the first position. The K values of 3 and 5 represent the search performance when the correct search result does not appear in the first position and when there are multiple correct search results. 1, 5, and 10 are also three parameters commonly used in the search system.

Figure 5 illustrates the comparison between the SCSPC method and the other three benchmark methods on the evaluation metric MRR. It can be seen that SCSPC has better performance on MRR than SWIM, QECK and CODEnn. SWIM and other methods ignore the semantics between code fragments, and SCSPC method performs diversified processing on data, which can search code more accurately.

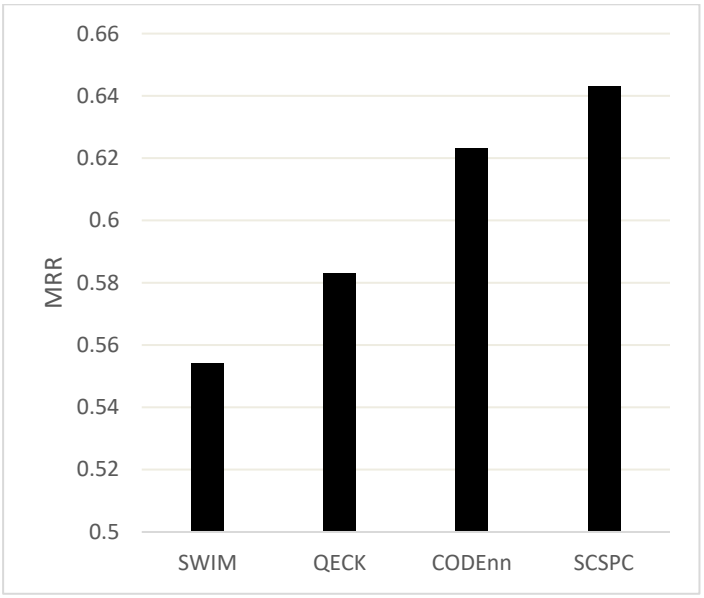

Figure 5. The comparison of the performance of the three methods.

Table 1 lists the comparison between the SCSPC method and the other three benchmark methods on the evaluation metric SuccessRate@K. It can be seen that the SCSPC method is better than the other three benchmark methods when k is 1, 5 and 10. The SCSPC method fully captures the semantic relationship between code snippets and natural language, and has a greater probability of correct answers in the search results.

Table 1. Performance comparison of three methods based on SuccessRate@k

Method	S@1	S@5	S@10
SWIM	0.546	0.558	0.594
QECK	0.553	0.634	0.664
CODEnn	0.549	0.647	0.693
SCSPC	0.641	0/693	0.734

5. Conclusion

In this paper, we propose SCSPC, a semantic code search method based on program transformation. This method performs data augmentation on Java code snippets in CodeSearchNet, and performs program transformation on five different aspects of the code while maintaining the semantic unchanged to obtain a large code corpus of high quality. Then, the CodeBERT model is fine-tuned to generate sentence vectors for natural language and code snippets to complete code search. This method deals with data diversification and effectively improves the accuracy of search results. In the following research on code search, we can improve the accuracy of the results from all aspects of the code snippet.

The method in this paper is to search in the case of specified programming languages, and future work can expand the scope of programming languages to further mine other semantic information of the code and improve the search accuracy.

6. Acknowledgments

This paper is supported by the following projects: Research on software code reuse and automatic generation(2020YFG0299).

7. References

[1] Lv F, Zhang H Y and Wang S W 2015 CodeHow: Effective Code Search Based on API Understanding and Extended Boolean Model *30th IEEE/ACM International Conference on Automated Software Engineering (ASE). ACM*, 2015:260-270.

[2] Lu M L, Wang S W and Lo D 2015 Query expansion via WordNet for effective code search *22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER). IEEE*, 2015:545-549.

[3] Lucr D 2004 A survey on software components search and retrieval, Proceedings *30th Euromicro Conference*, 2004:152-159.

[4] Keivanloo I, Rilling J, Zou Y 2014 *Spotting working code example.*

[5] Devlin J, Chang M W and Lee K 2019 BERT: pre-training of deep bidirectional transformers for language understanding *Proc of Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Stroudsburg, PA: Association for Computational Linguistics,* 2019: 4171-4186

[6] Vaswani A, Shazeer N and Parmar N 2017. Attention Is All You Need *arXiv*, 2017:5998-6008

[7] Feng Z, Guo D and Tang D 2020 CodeBERT: A Pre-Trained Model for Programming and Natural Languages, 2020:1536-1547.

[8] Husain H, Wu H and Gazit T 2019 CodeSearchNet Challenge: Evaluating the State of Semantic Code Search [EB/OL]. (2019-9-24)[2022-2-21]. *http://arxiv.org/abs/1909.09436*.

[9] Raghothaman M, Wei Y and Hamadi Y 2017 SWIM: Synthesizing What I Mean[J]. IEEE, 2017:357-367.

[10] Nie L, He J, and Ren Z 2017 Query Expansion Based on Crowd Knowledge for Code Search[J]. *IEEE Transactions on Services Computing*, 9(5):771-783.