An approach for code suggestion via code search

Divya Kumari Tankala [1], Dr.T.Venu Gopal [2]
[1]Department of CSE, G.Narayanamma Institute of Technology and Science (for women), Hyderabad, Telangana, India.
[2]Professor, JNTUHCE, Rajanna Siricilla, Telangana, India

Abstract-Programmers frequently write code that can be similar to existing code which may be written somewhere. An approach could help programmers to complete partially written code snippets to implement necessary functionality, help to discover extended lines of code to the partial code which are commonly used statements by other programmers, help to cross-check against similar code written by other programmers and also helps in fixing common mistakes and errors. Although there are many existing techniques which could potentially be used to get code suggestions. For example, code-to-code search tools could retrieve relevant code snippets from a corpus using a partial code snippet as query. However, the challenging task is to locate relevant source code from the large size code repositories and returns lots of relevant code snippets without removing or aggregating similar-looking ones. So, a new approach for code suggestion can be developed which takes a code snippet (partial code) as input and which can recommend several succinct source codes. The tool performs manual filtering from the dataset available and predicts the top five similar codes based on the similarity score.

Index terms: Code recommendation, code suggestion, clone detection, feature based code representation, auto code completer, code search, code snippet

1. INTRODUCTION

Deep learning systems now excel in most of the domains like software engineering, health domain etc. while the programmers wite the logic, if the systems help in auto completion of logic or some similar code snippets suggested with respect to the code entered as a query can boost the productivity of programmer. There are many existing systems supports such mechanism with IDE (Integrated development environment) for specific language. But still there are challenges in recommending code snippets for partial code written by programmer. Suppose a java programmer wants to write code to read a file or string then the developer is familiar with the libraries necessary to write the code, but they are not quite sure how to write the code properly to implement functionality with appropriate error handling and suitable configurations. They write code snippet shown in Figure 1 as an attempt. The programmer wants to know the customized way to extend the code so that the logic gets completed to do specific functionality of program. Common errors or exceptions can be handled.

```
InputStreamReader input = new InputStreamReader(system.in);
BufferedReader b = new BufferedReader(input);
```

Figure 1 Suppose a Java programmer writes this code to read arguments

```
FileReader file = new FileReader(String file);
BufferedReader buffer = new BufferedReader(file, int size);
```

Figure 2 A code suggestion to create a BufferdReader with specified size internal buffer

```
try {
        FileReader file = new FileReader("file.txt"); // Creates a FileReader
        BufferedReader i = new BufferedReader(file); // Creates a
BufferedReader
             i.read(array);
             System.out.println(array);
        i.close();
  }
  Catch (IOException e) {
      e.getStackTrace();
  }
```

Figure 3 Another suggested code snippet that shows how to properly close the buffer reader and handle any potential IOException.

There are few techniques for searching of codes, an approach to define a grammar, which enumerate all derivations of the grammar, checking each one for consistency with the examples. This approach can be combined with pruning based on types and other logical reasoning (Feser et al., 2015) [1]. Another category of systems is based on Satisfiability Modulo Theories (SMT) solving. SMT with SAT-style search with theories like arithmetic and inequalities. Many program synthesis engines based on SMT solvers exist, e.g., Sketch (Solar- Lezama, 2008) [2] and Brahma (Gulwani et al., 2011) [3]. Sirres, et.al., (2018) [4] Proposed an approach COCABU to address vocabulary mismatch problem while search for code in the repositories, focuses to automatically expand developer code search queries (i.e. free-form queries) to retrieve relevant code elements. It is built from GitHub and Q&A posts from Stack OVERFLOW to find the most relevant source code examples for developer queries. Cocabu builds snippet index and code index to speed up the performance. User queries are augmented to accelerate search process by adding relevant API names, class or method names since search queries may not include them in general by developers, which lead to get less accurate search results. The current implementation of COCABU uses the scoring function implemented in the

Lucene library. This function combines the Boolean Model (BM) and the Vector Space Model (VSM) to determine the relevancy of a document given for a user query. Apart from these search techniques, we analyze the search engines for code. e, code-to-code search tools [5][6] could retrieve relevant code snippets from a corpus using a partial code snippet as query. However, such code-to-code search tools return lots of relevant code snippets without removing or aggregating similar-looking ones.

2. Motivation

Code suggestion or recommendation reduce memory usage and increases the productivity of programmer. Code suggestions can be provided from different clustering methods to filter manually to complete the partial code. In case of code snippets mentioned in figure 1, 2, 3 suggested reading a file, string, or any argument, closing input stream and to handle exceptions (if any) etc. So that the programmer can choose one of them to implement the functionality.

The proposed approach is based on the idea that new code often resembles code that has already been written somewhere. Therefore, programmers can benefit from recommendations from existing code. To substantiate this claim, we conducted an experiment to measure the similarity of new code to existing code. This experiment was conducted on a large codebase in the Hack language. The results are ranked by similarity score: the percentage of features in the search query that are also found in the search result. For each changeset, we took the top-ranked method and its similarity score. The challenging task is to search for similar code in large code corpus with a code snippet as a query. So that the goal of the approach is to find all similar code fragments to complete partial code and shows only top 5 code suggestions for the given code query. Few code representations are recommended based on feature-based matching or ranking of the code retrieved and clustering methods can also be used.

2.1 Few existing models and disadvantages

Code to code search tools: Return lots of relevant code snippets without removing or aggregating similar-looking ones.
Pattern- completion: Cannot recommend any code outside the mined patterns.
Code clone detectors: Retrieve code snippets that are almost identical to a query snippet.

3. Proposed Methodology

Proposed model is a code recommendation tool which takes input a source code snippet i.e., a partial code. From the dataset available, the tool is supposed to recommend similar succinct source codes based on the similarity measure. Proposed model is expected to perform filtering among the available source codes present in the dataset. This would save the time of a software developer from searching across the Google until a suitable relevant code is found. The major objectives are

- Can make a search query with the code snippet itself.
- Fast enough to use in real time and to create recommendations within seconds even for very large codebases.
- A code snippet recommended does not simply come from a single method body, but is generated from several similar-looking code snippets via intersection.

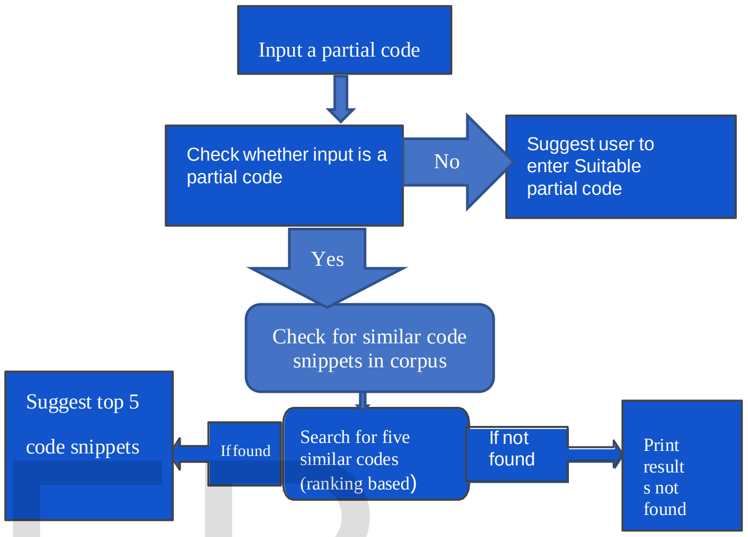
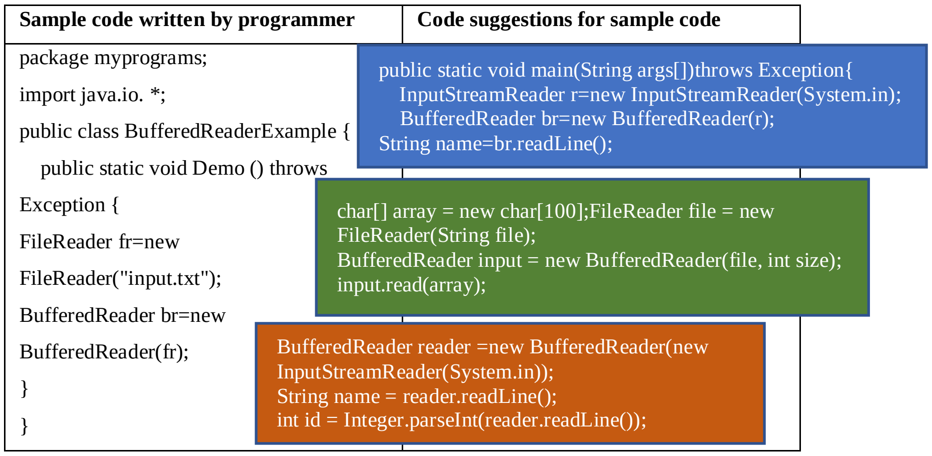

Input a partial code
Check whether input is a partial code
No
Suggest user to enter Suitable partial code
Yes
Check for similar code snippets in corpus
Suggest top 5 code snippets
If found
Search for five similar codes (ranking based)
If not found
Print results not found

Figure 4: Block diagram of proposed methodology for code suggestions

The tool takes a partial code snippet as input and checks whether the given input is valid or not. If the input is not a partial code, then the tool is expected to give a suggestion or recommend to enter only for a partial code. Now the tool looks up for the similar codes relevant to the partial code given as input and recommends top five codes based on the similarity measure. If they are not present in the repository then it would print no results found too to the user as shown in Figure 4. Based on code snippet entered at left hand side of figure 5, three code snippets are represented as code suggestions to read a file or to read command line arguments for java program. Try- catch statement also can be suggested along with code recommendations and also suggests code snippet to close byte stream or character stream classes for a given code snippet in java.

The proposed approach searches for a small set (e.g. 1000) of method bodies which contain the query code snippet approximately. A challenge in designing this search step is that a query snippet, unlike a natural language query, has structure, which should be taken into account while searching for code. It then ranks the retrieved code snippets based on the similarity with query snippet. After retrieving the similar code snippets, it groups them as cluster, which fall under same cluster so that it reduces redundancy of code fragments. Then the query snippet is compared against cluster and retrieves code fragments from that respective cluster and suggests them based on ranking. This approach can help better to boost the productivity of programmer and saves lots of time. This approach implemented using python and BigCloneBench dataset which contains Java Programs and it is evaluated with precision and recall metrics. If it is extended to large set of multi-language dataset, we can get better results using this approach.



Sample code written by programmer	Code suggestions for sample code

```
package myprograms;
import java.io. *;
public class BufferedReaderExample {
    public static void Demo () throws
Exception {
FileReader fr=new
FileReader("input.txt");
BufferedReader br=new
BufferedReader(fr);
}
}
```

```
public static void main(String args[])throws Exception{
    InputStreamReader r=new InputStreamReader(System.in);
    BufferedReader br=new BufferedReader(r);
String name=br.readLine();
```

```
char[] array = new char[100];FileReader file = new
FileReader(String file);
BufferedReader input = new BufferedReader(file, int size);
input.read(array);
```

```
BufferedReader reader =new BufferedReader(new
InputStreamReader(System.in));
String name = reader.readLine();
int id = Integer.parseInt(reader.readLine());
```

Figure 5: A sample screen to show the code recommendations for given code query

4. Conclusion

The proposed approach works based on ranking and clustering techniques, which able to retrieve similar code fragments using similarity score over the millions of java programs. If it is extended to large set of multi-language dataset, we can get better results using this approach. For a given code snippet query the proposed model searches in the corpus for method bodies containing the input snippet and outputs the top 5 recommendations based on similarity score. Though new code is frequently similar to existing code in a repository, currently available code search tools do not leverage this similar code to help programmers add to or improve their code, which would save the time of a software developer from searching across the Google until a suitable relevant code is found.

References

[1] John K. Feser, Swarat Chaudhuri, and Isil Dillig. Synthesizing data structure transformations from inputoutput examples. In Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI), 2015.

[2] Armando Solar-Lezama. Program Synthesis by Sketching. PhD thesis, EECS Dept., UC Berkeley, 2008.

[3] Sumit Gulwani, Susmit Jha, Ashish Tiwari, and Ramarathnam Venkatesan. Synthesis of loop-free programs. In Proceedings of the 32nd ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI), 2011.

[4] Sirres, R., Bissyandé, T.F., Kim, D. et al. Augmenting and structuring user queries to support efficient free- form code search. Empir Software Eng 23, 2622–2654 (2018).

[5] Kisub Kim, Dongsun Kim, Tegawendé F. Bissyandé, Eunjong Choi, Li Li, Jacques Klein, and Yves Le Traon. 2018. FaCoY: A Code-to-code Search Engine. In Proceedings of the 40th International Conference on Software Engineering (ICSE '18). ACM, New York, NY, USA, 946–957. https://doi.org/10.1145/3180155.3180187.

[6] Ken Krugler. 2013. Krugle Code Search Architecture. Springer New York, New York, NY, 103–120. https://doi.org/10.1007/978-1-4614-6596-6_6