Search in Source Code based on Identifying Popular Fragments

Eduard KURIC*

Slovak University of Technology in Bratislava
Faculty of Informatics and Information Technologies
Ilkovičova 3, 842 16 Bratislava, Slovakia
kuric@fiit.stuba.sk

When programmers write new code, they are often interested in finding definitions of functions, existing, working fragments with the same or similar functionality, and reusing as much of that code as possible. Short code fragments, which are returned to a programmer's query, do not provide enough "background" to help them how to reuse the fragments. Keyword-based code search instruments (tools) face the problem of low precision on their results due to the fact that a single word of the programmer's query may not match the desired functionality. Understanding code and determining how to use it, is a manual and time-consuming process. In general, programmers want to find initial points such as relevant functions. They want easily understand how the functions are used and see the sequence of function invocations in order to understand how concepts are implemented. When programmers learn about a program (source code), the control flow (execution of function calls) needs to be followed. It means successive jumping from one function to another.

Our main goal is to enable programmers to find relevant functions to query terms and their usages. In our approach, identifying popular fragments is inspired by PageRank algorithm, where the popularity of a function is determined by how many functions call it. We designed a model based on the vector space model, by which we are able to establish relevance among facts, which content contains terms that match programmer's queries directly. Our method consists of two phases, namely processing of the source code repository and searching for relevant functions given a programmer's query (see Figure 1).

Index creator creates document and term indexes from the source code repository. The function graph creator creates a directed graph of functional dependencies. The PageRank process is run on the directed graph of functional dependencies. It calculates a rank vector, in which every element is a score for each function in the graph.

When a programmer enters a query (1), a list of relevant documents (source code files) is retrieved (2). The list contains documents, where at least one query term occurs

* Supervisor: Professor Mária Bieliková, Institute of Informatics and Software Engineering

Spring 2012 PeWe Workshop, April 1, 2012, pp. 17–18.

in each document. Similarity (3) between two documents (query q and a relevant document d_j) is calculated (4) using the cosine distance.

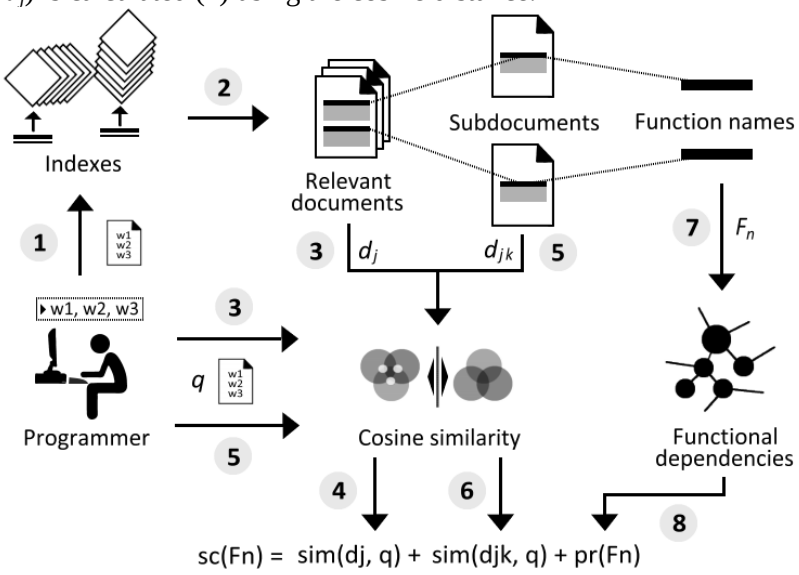

Figure 1. Searching for relevant functions.

Subdocument processing is as follows*:*

1. Each retrieved relevant document is divided into subdocuments, where each one contains only one definition of a function with surrounded comments (if any).
2. From each subdocument, terms are extracted from comments, function name and identifiers. For the terms, TF/IDF weights are calculated. The cosine similarity (5) is calculated between each subdocument d_{jk} and the query q (6).

Ranking of the relevant functions is as follows:

1. Names of defined functions are extracted from the relevant documents.
2. For each function name F_n, a final score $sc(F_n)$ is calculated as sum of:
   a. a similarity between the programmer's query q and the document d_j, in which is the function F_n defined (4); a similarity between the programmer's query q and the subdocument d_{jk}, in which is the function F_n defined (6),
   b. a PageRank score $pr(F_n)$ for the F_n (7)(8).

Acknowledgement. This work was partially supported by the Scientific Grant Agency of Slovak Republic, grant No. VG1/0675/11.

References

[1] Grechanik, M, et al.: A search engine for finding highly relevant applications. *In Proc. of the 32nd ACM/IEEE Int. Conf. on Softw. Eng.*, NY, 2010, pp. 475-484.

[2] Sillito, J., et al.: Asking and Answering Questions during a Programming Change Task. *IEEE Trans. Softw. Eng.*, vol. 4, 2008, pp. 434-451.