Search in Source Code Based on Identifying Popular Fragments

Eduard Kuric, prof. Mária Bieliková

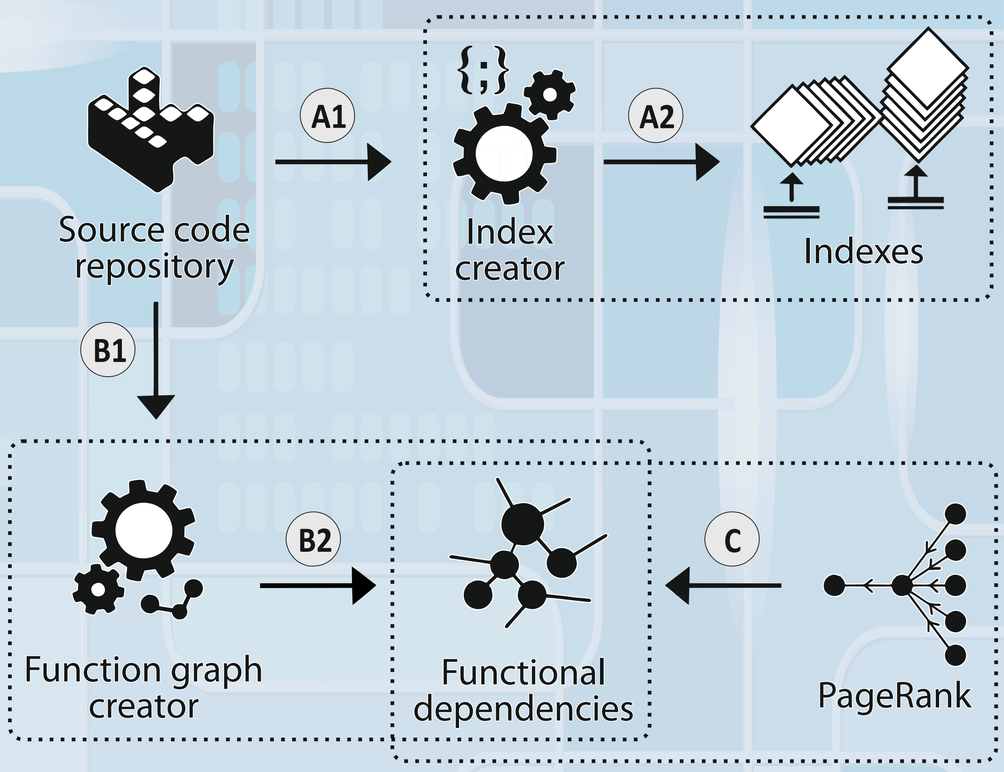
> Motivation

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. **They want easily understand** how the functions are used and see the sequence of function invocations in order to understand how concepts are implemented.

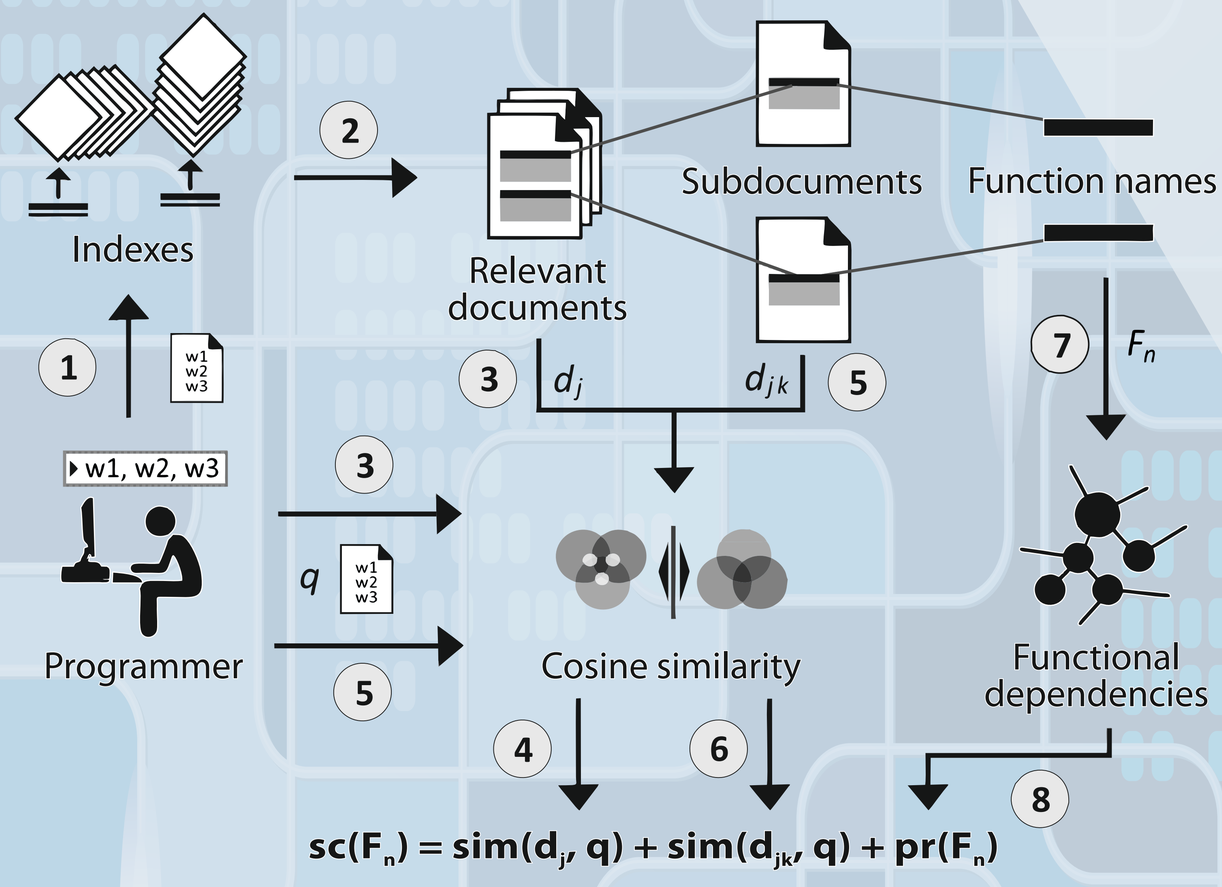
> Goals

enable programmers to **find relevant functions to query terms** and their usages; and see associations between concepts in the functions and query

> Processing source code repository



> Searching for relevant functions given a query



The index creator (A1) creates document and term indexes (A2).
- based on the vector space model - each document is modelled as a vector of terms, which occur in that document
- terms are extracted from comments, names of functions and identifiers -- NLP techniques are applied

The function graph creator (B1) creates a directed graph of functional dependencies (B2).
- nodes represent functions (names of functions)
- a directed edge between the function *G* and the function *H* is created if the function *H* is invoked in the function *G*

The PageRank process (C) is run on the directed graph of functional dependencies, and it calculates a rank vector, in which every element is a score for each function in the graph.
- the PageRank of a function is defined recursively and depends on how many functions call (invoke) it
- the rank value indicates an importance of a particular function

Retrieving the relevant documents:
1. for a query (1), a list of relevant documents (code files) is retrieved (2);
2. similarity (3) between the query *q* and a relevant document d_j is calculated (4) using the cosine distance;
3. each retrieved relevant document is divided into subdocuments, where each one contains only one definition of a function with surrounded comments (if any);
4. from each subdocument, terms are extracted from comments, function name and identifiers; for the terms, TF/IDF weights are calculated; and the cosine similarity (5) is calculated between each subdocument d_{jk} and the query *q* (6).

Ranking the relevant functions:
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 a sum of:
   a. a similarity between the query *q* and the document d_j, in which is the function F_n defined (4); a similarity between the 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).

> Method evaluation

- project Vilcacora for interactive browsing of multimedia content
- the precision metric - the fraction of the top 10 ranked functions relevant to a query; for each query a level of confidence was assigned (completely/mostly irrelevant, mostly/highly relevant)

Examples of the results for 3 queries (0.74 mean precision for 15 queries)

query	load, image, create, texture, map object	image, transform, flip, aspect, ratio	mask, texture, grayscale
precision	0.7	0.4	0.8

eMail: kuric@fiit.stuba.sk bielik@fiit.stuba.sk
web: www.fiit.stuba.sk/~kuric www.fiit.stuba.sk/~bielik

STU FIIT · it.src · PeWe@FIIT personalized web group · PerconIK Personalized Conveying of Information and Knowledge