Code Snippet Search Engine

3/18/2010

Phitchayaphong Tantikul (21325439)

Hye Jung Choi (38924967)

Table of Contents

1. Introduction

There are a lot of code search tools to serve different needs of users who look for source code. A group of code search tools mainly supports a search for open-source components or projects. Another group of code search tools provides code snippet search. The other group of code search tools aids users with looking for any of either open-source components/projects or code snippets. Although a lot of code search tools have been proposed for open-source component or project search, there is lack of support on code snippet search.

People mostly tend to look for code snippets by using general purpose text search engines like Google, Yahoo, and so forth. According to an empirical study, 84% of developers use general purpose search engines to look for source code [5]. In most cases, those search engines provide decent results so that people are unlikely to recognize the fact that code structure and other properties source code are ignored in search results. In other words, source code is treated as text, so source code is tangled with normal text and just looks like English sentences in search results. It is hard to read source code shown in search results. Therefore, users need to click on a link and go over a web page to find a structured source code.

On the other hands, code-specific search engines, such as Google Code Search[1], Krugle[2], or Koders[3], consider not fragments of source code, but a whole source file as an individual document for searching. In addition, returned results from these code-specific search engines only show source code themselves without any additional explanation about the source code. They sometimes provide comment lines along with code in search results. However, if no comment is available or comments are not enough to shortly explain what the code snippet is, users need to click a link and read a whole or part of a source file.

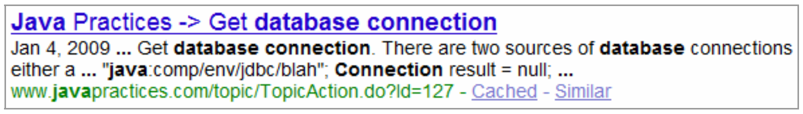
To address lack of supports on code snippet search, we would like to focus on code snippet search. Besides, we would like to develop a better way to present search results to users. Thus, we may be able to mitigate limitations of either current code search engines or general purpose search engines to look for source code. Our contribution is to make searchable repository using code snippets with relevant textual information that explains the details of each code snippet and to provide a better code search engine to users.

2. The gap between code search and web search engines

Problems of general purpose web search engines: As we briefly mentioned in the introduction, the problem of general purpose web search engine is that source code is treated as text so that a structure of code is ignored. For example, when we look for Java source code in Google, Google shows either a snippet of text (lines of the most relevant sentences) or some code. Figure 1 shows one of search results using a query, "*java connect to database*." This is one of top 10 results and

[1] http://www.google.com/codesearch
[2] http://www.krugle.com/
[3] http://www.koders.com/

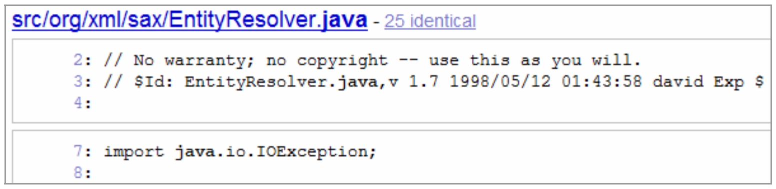
showing both text and code snippet under a page link. The code section is treated as a plain text so that we cannot know whether there is a code that we want at a glance.



Java Practices -> Get database connection
Jan 4, 2009 ... Get **database connection**. There are two sources of **database** connections either a ... "**java**:comp/env/jdbc/blah"; **Connection** result = null; ...
www.**java**practices.com/topic/TopicAction.do?Id=127 - Cached - Similar

Figure 1: Google result

Problems of current code search engines: The problem of current search engines that are specifically designed for code search is that they tend to return only source code without any textual explanation about code snippets in search results. Since each page is corresponding to each Java file, some result pages have very long source code. Although some code search engines locate more relevant code sections in search results, users may read and understand other sections of source code to find out what they really need. Users may need only a few lines of source code and may not want to read lengthy source code. On the other hands, returned code snippets sometimes do not seem relevant to our search query, or it is hard to have a clue to what returned code snippets mean to us. As an instance, Figure 2 represents a result from Google Code Search using the same keyword, "*java connect to database*." Given a snippet, we cannot know what the code means, especially without any comment. The only clue to understand a code snippet is a file name or an import statement used inside code. Good comments in the source code could give some idea about the source code to users, but a lot of source codes do not have useful comments in their source code. More seriously, the other problem is that when a user enters a free text query for searching, code search tools do not return better results than general purpose search engines to users. In fact, a lot of users input conceptual-level free text queries when they do not have specific idea about their implementation. Therefore, it is true that when users use free text queries to look for source code, general purpose search engines perform better. Contrarily, when users use Java API or implementation relevant queries, code search engines work better.



src/org/xml/sax/EntityResolver.java - 25 identical

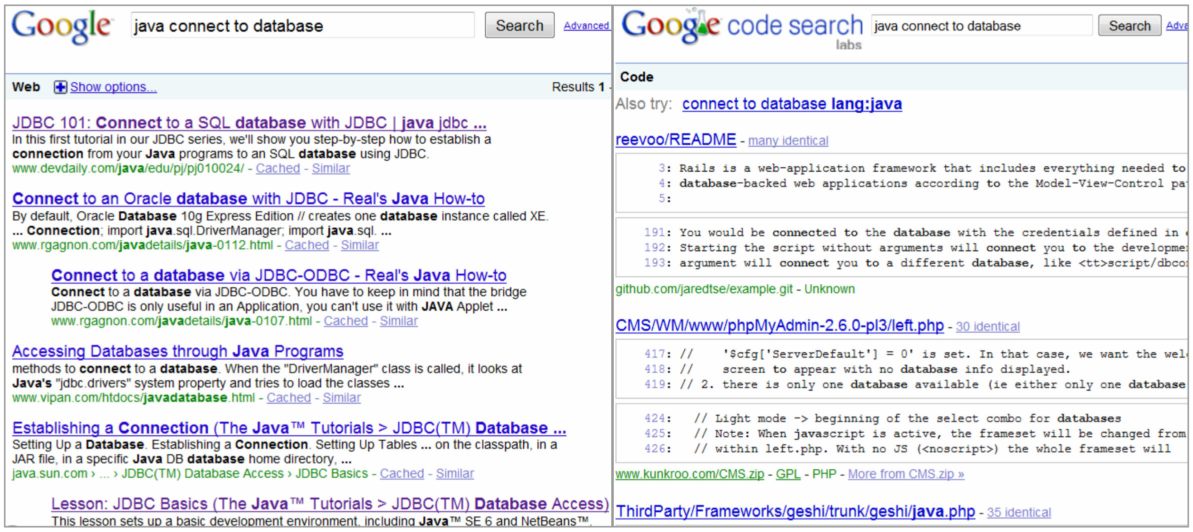
```
// No warranty; no copyright -- use this as you will.
// $Id: EntityResolver.java,v 1.7 1998/05/12 01:43:58 david Exp $

import java.io.IOException;

```

Figure 2: Google Code Search result

Figure 3 clearly shows results from both Google and Google Code Search using a same query, "*java connect to database*." Based on results from both engines, we evaluated which one provided better and more relevant results to us. We concluded that results from Google were more relevant to our needs when we used free text query. Results from Google Code Search were not much useful to us. Interestingly, when we used "*connect to database lang:java*" from a *Also Try* link that was recommended by Google Code Search, results were much better than the previous query in Figure 4. We assume that Google Code Search is working better when user queries are more similar to Java code, which is more formal and relevant to implementation details.



java connect to database

Web Show options... Results 1

JDBC 101: **Connect** to a SQL **database** with JDBC | **java** jdbc ...
In this first tutorial in our JDBC series, we'll show you step-by-step how to establish a **connection** from your **Java** programs to an SQL **database** using JDBC.
www.devdaily.com/**java**/edu/pj/pj010024/ - Cached - Similar

Connect to an Oracle **database** with JDBC - Real's **Java** How-to
By default, Oracle **Database** 10g Express Edition // creates one **database** instance called XE. ... **Connection**; import **java**.sql.DriverManager; import **java**.sql. ...
www.rgagnon.com/**java**details/**java**-0112.html - Cached - Similar

Connect to a **database** via JDBC-ODBC - Real's **Java** How-to
Connect to a **database** via JDBC-ODBC. You have to keep in mind that the bridge JDBC-ODBC is only useful in an Application, you can't use it with **JAVA** Applet ...
www.rgagnon.com/**java**details/**java**-0107.html - Cached - Similar

Accessing Databases through **Java** Programs
methods to **connect** to a **database**. When the "DriverManager" class is called, it looks at **Java's** "jdbc.drivers" system property and tries to load the classes ...
www.vipan.com/htdocs/**javadatabase**.html - Cached - Similar

Establishing a **Connection** (The **Java**™ Tutorials > JDBC(TM) **Database** ...
Setting Up a **Database**. Establishing a **Connection**. Setting Up Tables ... on the classpath, in a JAR file, in a specific **Java** DB **database** home directory, ...
java.sun.com › ... › JDBC(TM) Database Access › JDBC Basics - Cached - Similar

Lesson: JDBC Basics (The **Java**™ Tutorials > JDBC(TM) **Database** Access)
This lesson sets up a basic development environment, including **Java**™ SE 6 and NetBeans™.

code search labs — java connect to database

Code

Also try: connect to database **lang:java**

reevoo/README - many identical

```
Rails is a web-application framework that includes everything needed to
database-backed web applications according to the Model-View-Control pa

```

```
You would be connected to the database with the credentials defined in
Starting the script without arguments will connect you to the developme
argument will connect you to a different database, like <tt>script/dbco
```

github.com/jaredtse/example.git - Unknown

CMS/WM/www/phpMyAdmin-2.6.0-pl3/left.php - 30 identical

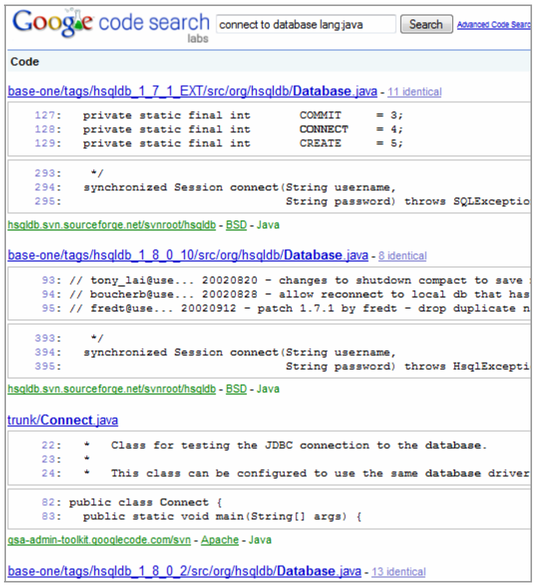
```
//    '$cfg['ServerDefault'] = 0' is set. In that case, we want the wel
//    screen to appear with no database info displayed.
// 2. there is only one database available (ie either only one database
```

```
  // Light mode -> beginning of the select combo for databases
  // Note: When javascript is active, the frameset will be changed from
  // within left.php. With no JS (<noscript>) the whole frameset will
```

www.kunkroo.com/CMS.zip - GPL - PHP - More from CMS.zip »

ThirdParty/Frameworks/geshi/trunk/geshi/**java**.php - 35 identical

Figure 3: Google vs. Google Code Search



code search labs — connect to database lang:java

Code

base-one/tags/hsqldb_1_7_1_EXT/src/org/hsqldb/**Database**.java - 11 identical

```
  private static final int       COMMIT     = 3;
  private static final int       CONNECT    = 4;
  private static final int       CREATE     = 5;
```

```
   */
  synchronized Session connect(String username,
                               String password) throws SQLExceptio
```

hsqldb.svn.sourceforge.net/svnroot/hsqldb - BSD - Java

base-one/tags/hsqldb_1_8_0_10/src/org/hsqldb/**Database**.java - 8 identical

```
// tony_lai@use... 20020820 - changes to shutdown compact to save
// boucherb@use... 20020828 - allow reconnect to local db that has
// fredt@use... 20020912 - patch 1.7.1 by fredt - drop duplicate n
```

```
   */
  synchronized Session connect(String username,
                               String password) throws HsqlExcepti
```

hsqldb.svn.sourceforge.net/svnroot/hsqldb - BSD - Java

trunk/**Connect**.java

```
 *   Class for testing the JDBC connection to the database.
 *
 *   This class can be configured to use the same database driver
```

```
public class Connect {
  public static void main(String[] args) {
```

gsa-admin-toolkit.googlecode.com/svn - Apache - Java

base-one/tags/hsqldb_1_8_0_2/src/org/hsqldb/**Database**.java - 13 identical

Figure 4: Google Code Search using a *lang:* Operator

The gap between the code search engine and the general web search engine: Free text queries at the conceptual-level are more common and natural to users than those based on semi-formal or formal programming languages to look for source code on the web. When users look for source code, they often do not know specific details of source code, such as names of API or components, or implementation details. Therefore, an effective code search engine should support free text queries and provide good results to users. Currently, neither code search engines nor general purpose text search engines support code search effectively. In order to facilitate code search, conceptual gap [1-2, 4] between problem domain and solution domain must be mitigated, and we need a better code search engine that could bridge the gap.

3. The way to mitigate the gap

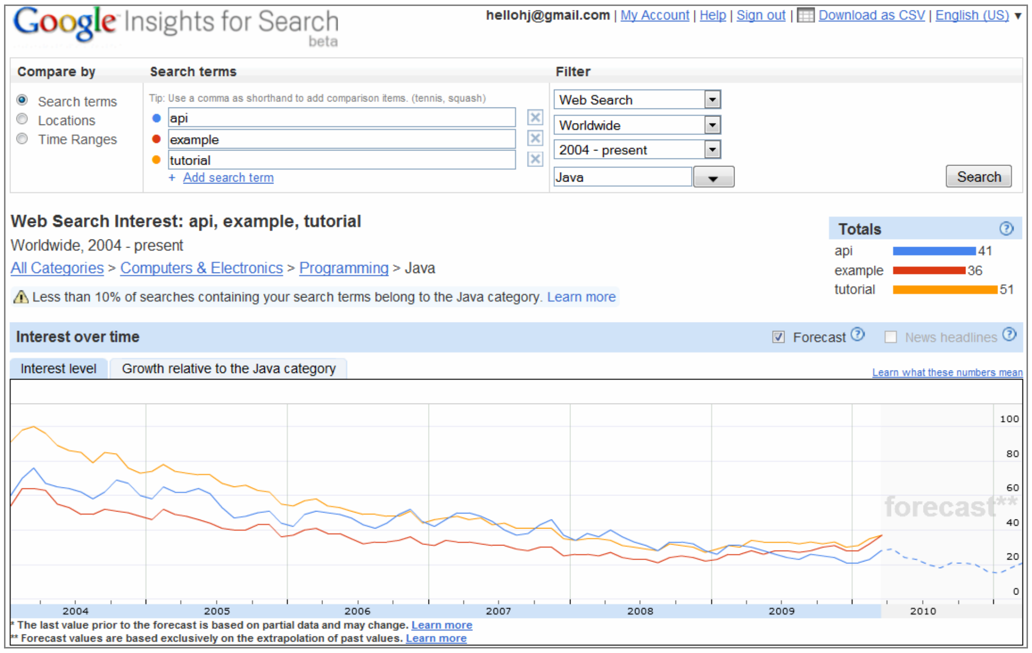
We aim to find a way to mitigate the gap between code search engines and general web search engines. Good things that we want to take from general purpose web search engines are free text queries and text snippets (or summary) explaining each web page in search results. A good thing from code search engines is that they show source code with its original structure so that users can read source code easily. Our goal is to build a code snippet search engine that users enter free text queries to look for source code and search results are displayed with both useful and relevant text information and structured code snippet.

A more detailed scenario of our code search engine is that both texts and code snippets are indexed and ranked for searching. Users enter free text queries to look for source code. Then, most relevant text information and code snippet with an original structure are presented together in the search result page. Therefore, users may be able to determine which page is more relevant to their needs with ease. Besides, code snippets may reduce the time for developers to decide whether the search result is helpful because it can show only the point where users are interested in. A pair of good code snippet and brief explanation in the search results may provide enough information meeting users' needs so that users may not need to click on the result and stop searching. In other words, we may be able to achieve good abandonment [3] by providing relevant code snippets and text explanation to users.

To provide both text and code snippet under each link in search results, we concentrate on tutorial web pages containing Java code snippets and text explanation about code. According to the result from Google Insight for Search[4] in Figure 5, it seems that tutorial has been one of the most popular means to look for source code under Java programming language in Google. This implies that a lot of users are more interested in tutorial pages while looking for information on Java programming language rather than API, components, or examples that may or may not contain only code.

Motivation: A lot of tools support API or component search, or open-source project search. However, there has been lack of support in code snippet search. We wonder developers' actual interests among API or components, tutorial, or example when they search for source code. To know the trend of specific search terms under Java programming, we utilized Google Insights for Search. Our selected search terms are ***api***, ***tutorial***, and ***example***. To filter results, we chose Web Search, Worldwide, 2004-present, and Java programming among various options. In Figure 5, the blue line represents API, the red line is for example, and tutorial is an orange line. The result is normalized and presented on a scale from 0-100. According to Google Insight, each point on the graph is divided by the highest point, or 100. When there is not enough data, 0 is presented. In Figure 5, less than 10% of searches containing search terms belong to the Java category. Regarding forecast, Google predicts that API search may decrease more than other search terms.

[4] http://www.google.com/insights/search


Google Insights for Search beta

hellohj@gmail.com | My Account | Help | Sign out | Download as CSV | English (US)

Compare by: [✓ Search terms] Locations Time Ranges

Search terms: api, example, tutorial

Filter: Web Search; Worldwide; 2004 - present; Java

Web Search Interest: api, example, tutorial

Worldwide, 2004 - present

All Categories > Computers & Electronics > Programming > Java

Less than 10% of searches containing your search terms belong to the Java category. Learn more

Totals	
api	41
example	36
tutorial	51

Interest over time — [✓ Forecast] News headlines

* The last value prior to the forecast is based on partial data and may change. Learn more

** Forecast values are based exclusively on the extrapolation of past values. Learn more

Search
Tip: Use a comma as shorthand to add comparison items. (tennis, squash)
Add search term
Interest level
Growth relative to the Java category
Learn what these numbers mean
forecast**
100
80
60
40
20
0
2004
2005
2006
2007
2008
2009
2010

Figure 5: Google Insight for Search Result using three terms

Based on results from Google Insight for Search, we cannot assure that users prefer to look for tutorial for Java code search since Google mentions that less that 10% of searches might represent our concern. However, we can argue that although a lot of users have been looked for tutorial than API over time, there has not been evident and explicit support for code search on Java tutorial. Our prototype may be able to satisfy users' actual needs on code search by mitigating the gap between current code search tools and general purpose web search engines.

4. Prototype Code Search Engine

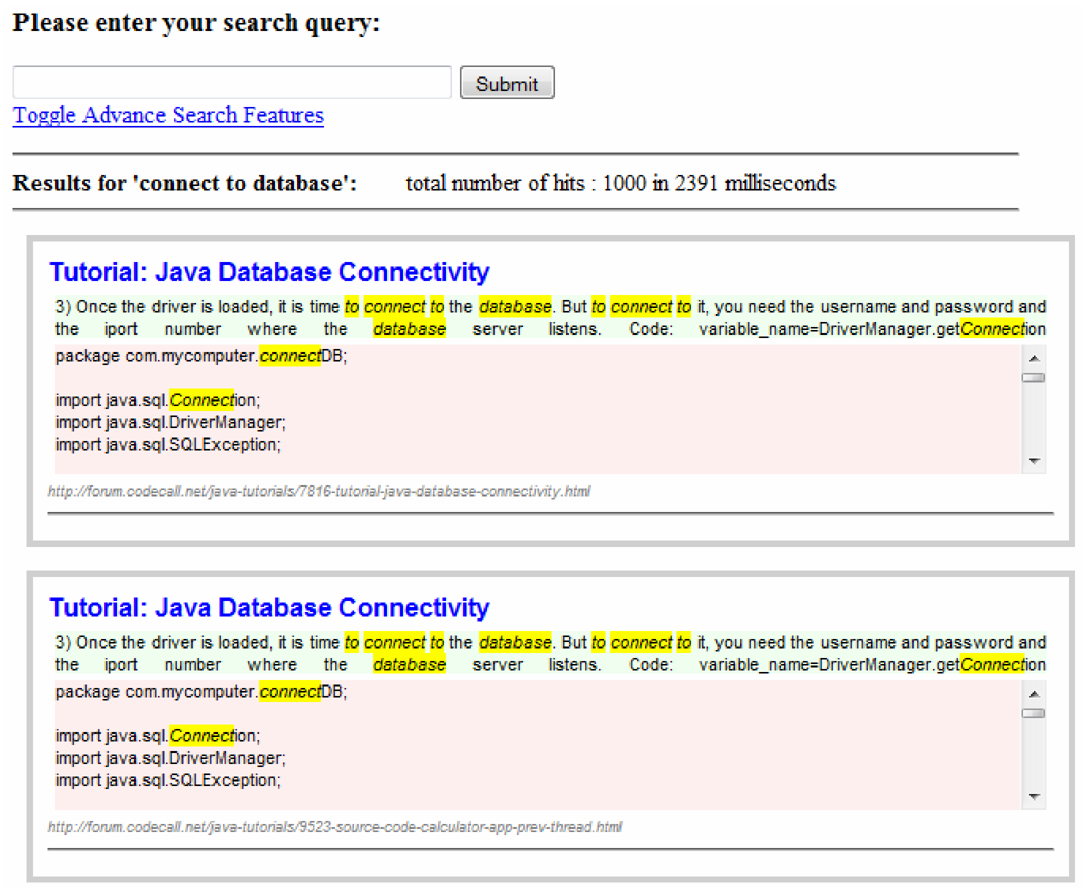
Our prototype code search engine is specifically designed for code search on Java tutorial pages. Since our research group has been working on code search, we can get a list of Java tutorial web sites and use them as our seed URLs for crawling. After crawling all tutorial pages from seed websites, we extract code snippets and text information from each page and store them in the MySQL[5] database. We also get this extraction module from our research group and modify it slightly for the purpose of our code search. Then, we filter out pages with non-Java source code and pages without any code snippets.

After all filtering processes, our prototype maps the most relevant text and code snippet in each page, and mapped pairs of text and code snippet are stored in another table of the database. All data is indexed and ranked using Lucene[6]. When a user enters a query in our Web interface, the prototype code search engine returns relevant search results with short description and code snippets. Figure 6 is a screenshot of our prototype code search engine[7].

[5] http://www.mysql.com/

[6] http://lucene.apache.org/

[7] http://peony.ics.uci.edu:8080/CodeSnippetRepository_Web/index.jsp



Please enter your search query:

Results for 'connect to database': total number of hits : 1000 in 2391 milliseconds

Tutorial: Java Database Connectivity

3) Once the driver is loaded, it is time to connect to the database. But to connect to it, you need the username and password and the iport number where the database server listens. Code: variable_name=DriverManager.getConnection

```
package com.mycomputer.connectDB;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
```

http://forum.codecall.net/java-tutorials/7816-tutorial-java-database-connectivity.html

Tutorial: Java Database Connectivity

3) Once the driver is loaded, it is time to connect to the database. But to connect to it, you need the username and password and the iport number where the database server listens. Code: variable_name=DriverManager.getConnection

```
package com.mycomputer.connectDB;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
```

http://forum.codecall.net/java-tutorials/9523-source-code-calculator-app-prev-thread.html

Figure 6: Prototype Code Search Engine

Figure 7 depicts the overall architecture of our prototype code search engine. We define six major steps for our prototype code search engine: (1) crawling, (2) code extraction, (3) matching between code and text, (4) building code snippet repository, (5) indexing, (6) ranking, and (6) searching. More technical details on the design of the prototype code search engine are described in the next section.

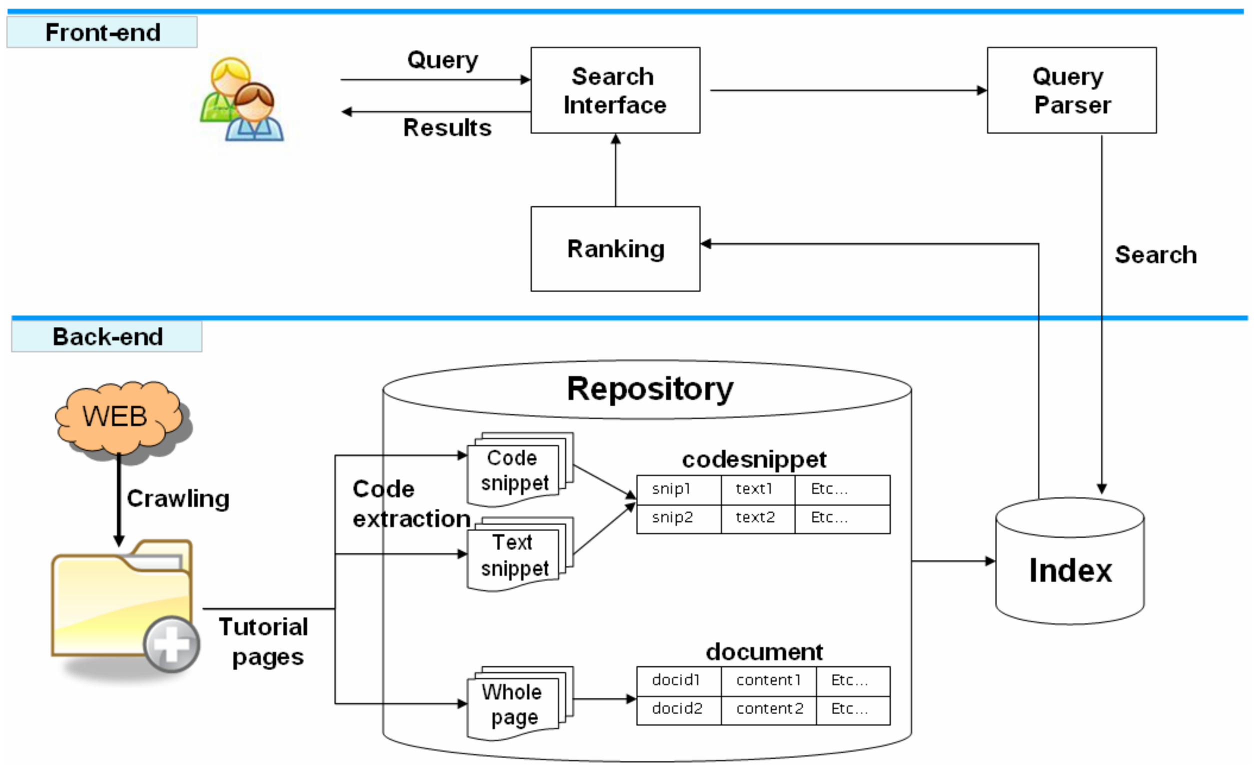


Figure 7: Design of the prototype code search engine

5. Design of the Code Search Engine

In this section of the report, we explain the design of the code search engine. The major parts we cover are (1) crawling, (2) code extraction, (3) matching, (4) code repository, (5) indexing, (6) ranking, and (7) searching.

Crawling: We started from crawling all pages from 33 seed tutorial websites. The list of websites we crawled is provided in the section 7, Additional Data. Although seed websites are identified as Java tutorial sites, we found that a lot of them included tutorials of other programming languages, such as VB.Net, C/C++, JSP, PHP, and so forth. Since our goal is to focus on Java tutorial pages, we excluded URLs that contained words signifying other programming languages in their URLs and downloaded all pages in the local file system. From 33 tutorial websites, the number of downloaded pages is 34,054.

Code extraction: Code extraction was conducted using a module that was developed in our research group. The algorithm used in code extraction is a combination of both the heuristics to detect <pre> tags in pages and machine learning. While reading each downloaded tutorial page, our extraction module identified code sections and text sections, and those results were stored in the database.

During the process of code extraction, we discovered a lot of things to consider. (1) The first thing is that a lot of pages are tutorials for other programming languages, so include code snippets of non-

Java languages. (2) The second thing is that a lot of pages contain only text information without any code snippet.

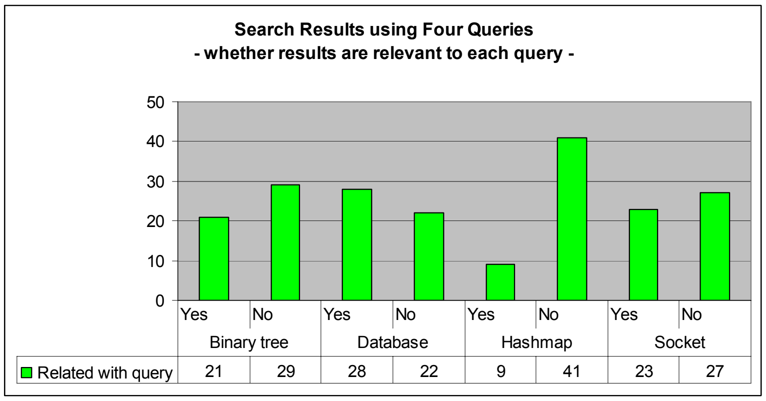
To solve the first problem, we defined a set of Java keywords and Java common classes from the java.lang package and counted their occurrences in code snippets. If no Java related term was found in the snippet, we assumed that the code snippet was unlikely to be Java source code. In fact, this simple heuristic worked really well. Using our approach, we could detect a lot of documents that did not contain Java source code. In addition, when code snippets in pages were so general or common that we could not determine their language, those pages were excluded from our consideration. To solve the second problem, we selected pages with only text and excluded those pages when we built code snippet table that is for pairs of code snippets and texts. Using our heuristics, we removed 21,162 pages. 12,823 pages were remained for code snippet search. The result is shown in Table 1.

Total downloaded pages	34,054
Pages with no code snippet & pages with non-Java source code	21,162
Pages after all filtering	12,823

Table 1: The summary of numbers of pages

Matching between code snippet and text: The biggest hurdle in this project is to determine which text section is more relevant to a code snippet. Four possible cases for relevance between a code snippet and a text are: text above a code snippet, text below a code snippet, text around a code snippet (both above and below), and no available text. To solve this issue, we conducted an empirical study and found that text above a code snippet was more relevant to a code snippet. For the study, we chose four different queries, such as *binary tree*, *database*, *hashmap*, and *socket*. Then, we got 200 pages (50 pages for each query) based on full-text search using MySQL, and search results were sorted on TF/IDF scores. First, we identified whether a result was relevant to a query. Second, if a page was identified as the relevant page to a query, we examined the location of related text around code snippet.

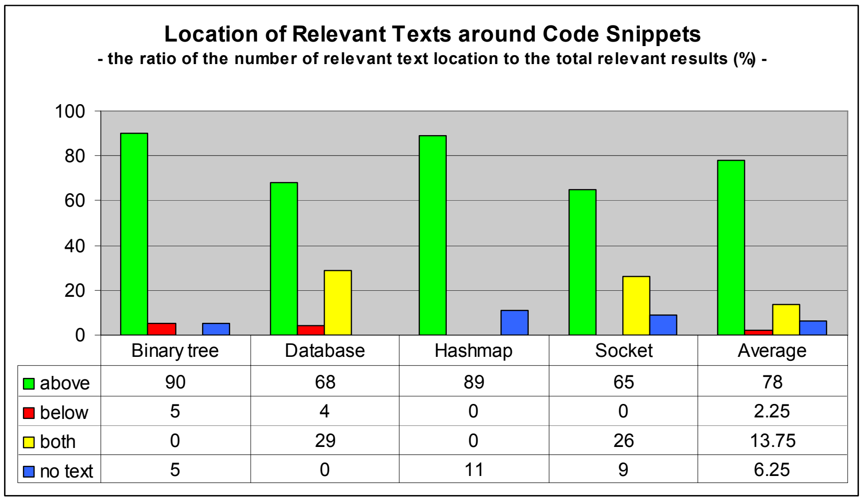
On average, 20.25 pages out of 50 pages were relevant to each query, which was 40.5% of each result. 29.75 pages were irrelevant to each query, which was 59.5% of results. After some investigation, the reason of this lower relevance between search results and queries was that some queries like *hashmap* and *socket* were so common that a lot of code snippets had hashmap or socket regardless of topics in pages. In Figure 8, we can easily see that 41 results out of 50 results are irrelevant to a query in case of hashmap. Similarly, 27 returned results of socket are irrelevant to a query. On the other hands, a lot of results of binary tree turned out irrelevant. The reason of this case was that each keyword of a query, *binary* and *tree*, was separately considered in search. Although top 15 results were very relevant to binary tree, other results were related with either binary or tree. Throughout analysis, we concluded that normal TF/IDF was not good for searching so that we decided to use TF normalization to avoid length bias. The detail of TF normalization is explained in Ranking below.



	Binary tree		Database		Hashmap		Socket	
	Yes	No	Yes	No	Yes	No	Yes	No
🟩 Related with query	21	29	28	22	9	41	23	27

Figure 8: Result relevance to each query

After identifying relevant pages against each query, we investigated where a relevant text was located around a code snippet. In the relevance check previously, 81 pages out of 200 total pages were selected as relevant pages. We checked these 81 relevant pages and determined the location of related texts around code snippets. The result is represented in Figure 9. The first four groups of bar graphs represent results from four queries, and the last group of bar graph shows an average of all results. On average, text above a code snippet is more relevant than others in 78% of cases: 2.25% for text below; 13.75% for text in both above and below a code snippet; no text is 6.25%. As a result, we decided our heuristic to determine relevance between text and code snippet for the project; text above a code snippet is more likely to explain the code snippet.



	Binary tree	Database	Hashmap	Socket	Average
🟩 above	90	68	89	65	78
🟥 below	5	4	0	0	2.25
🟨 both	0	29	0	26	13.75
🟦 no text	5	0	11	9	6.25

Figure 9: Location of relevant text around code snippets

Code snippet repository: In our code snippet database, we created two tables. One table is for original pages, and the other table contains pairs of code snippet and text extracted from the first table. For the first table (from now, we call it as document table), each row in the table contains all information of each page. According to our heuristic above, we paired code snippets and its text above it together and stored them in the second table (from now, we call it as code snippet table). Since one page may contain more than one code snippets, the number of records in the code snippet table is always equal to or greater than the number of records in the document table.

Indexing: All data in the repository are indexed using Lucene. Since there are two types of data, code snippet and text, in the table, we needed to use two approaches to index the data. For the text, we decided to split it into terms by using non-word characters. Then, we conducted normal indexing on those terms. On the contrary, code snippet part was more complicated because splitting words using the same method as we did for text would destroy its structural information. In order to keep that information, we developed a Java parser to extract all identifiers in code snippets and stored them in separate categories. The categories are “package”, “import”, “class declaration”, “class used”, “class extension and implement”, “method declaration”, “method invocation”, “variable declaration”, “return type” and “comment”. All those terms were indexed and used for searching and ranking. We also considered camel-case[8] terms, because it is common that developers use camel-case to put separate terms into one meaningful identifier for classes, methods, etc. Therefore, we also wrote a camel-case extractor to split those terms and used them for creating index to increase possibility that identifiers were found in search results.

Additionally, we also filtered out all the stop words from both code snippets and texts before indexing them. English stop word list[9] was used to filter out the common words that do not give any significant meaning to the text, such as ‘a’, ‘an’, ‘the’, ‘while’, ‘on’, and ‘at’. Similarly, the list of Java keywords, such as ‘class’, ’for’, ’new’, and ’void’, was used to filter unnecessary terms from code snippets.

Ranking: In the previous Matching section, we found a problem that was caused from the length of document. For example, we can see in Figure 8 that queries using hashmap and socket provide a lot of irrelevant results to users because keywords are commonly used in a lot of Java source code. We decided to normalize TF values to prevent this problem, using Lucene ranking which includes TF normalization (NTF).

To calculate ranking of search results, we defined our ranking algorithm. As mentioned above, we have two tables that contain different data; one is document table, and the other is code snippet table. For our ranking, we treat each table as a different corpus for ranking. Thus, two corpora are ranked by NTF/IDF individually. In our experiment in searching for same keyword in those tables separately, we found that searching in document table gave more relevant results than searching in code snippet table. While our expected output from this search engine was a list of code snippets, searching only in document table was not satisfactory to our goal. Therefore, we decided to improve search results from code snippet table by weighting their relevancy with score from document they came from. Finally, we decided to use multiplication between NTF/IDF values from both corpora to be our ranking scores. Since the NTF/IDF score was not normalized, its value could be greater than or less than one and it would cause errors while calculating scores if we multiply them. Therefore, we normalized the scores before multiplying them, by dividing them with maximum score from the best result from each table to make scores from each side range between 0 and 1.

In code snippet table we used 3 fields that we think they are most relevant that can be used for search. They are title of document, code snippet, and text above code snippet. Since we had three fields to search but we wanted only one single value to be score of this code snippet, we combined

[8] http://en.wikipedia.org/wiki/CamelCase
[9] RANKS.NL: http://www.ranks.nl/resources/stopwords.html

the scores from each field together using weights. After trying some combination of the weight, we equally weighted all those three fields.

Searching: We make a web user interface to look for code snippets. Our web server is Tomcat 5.5.9, which is located in the server in our laboratory. When a user enters a free text query in our search engine, we check relevance between the query and our code snippet repository and provide search results to the user. Our search results provide lines of the most relevant text and code snippet so that the user can easily know which result is the best matching to their needs.

Not only free-text query input, but in this project, we also provide keyword searching on various types of identifiers that we have indexed them in different fields, such as “class name”, “class used”, “method name”, “method invoked” and “variable name”. Users can perform a search in these field directly using special keyword such as “class:”, “classused:”, “method:” and “variable:” placed in front of search terms. For example, users can query “class:HashMap” to search for a declaration of a class called HashMap in any code snippet in repository.

6. Future Work and Conclusion

The current heuristic to identify Java source code in web pages is using Java keywords and Java common classes from the java.lang package. It worked really well when we looked at results before and after filtering. However, if time is allowed, we would like to implement more sophisticated Java syntax checker to identify Java source code. Since we filter our results using Java keywords and the most basic identifiers of Java, we cannot detect some Java source code when a code snippet has implementation specific details. For example,

IPackageFragmentRoot srcFolder = project.getPackageFragmentRoot(folder);

In this case, our prototype using the heuristic cannot detect automatically whether this code snippet is Java code or not so that it may be treated as non-Java and excluded from search results. However, a developer, a human inspector, may be able to know this snippet is Java code because IPackageFragmentRoot is Java-specific. Therefore, we need a better way to identify Java source code.

Another thing we are concerned with is how well our prototype may deal with scalability on the web. Since we have only 12,823 pages to search in our repository, we wonder the performance of our prototype when there are several millions of pages to search. Moreover, we consider only tutorial pages from 33 seed websites. In the future, we would like to include other web pages that have useful Java code snippets, such as forums, technical blogs, educational websites, and so on, so that users may be able to get better results from our search engines on much richer information.

The other possible work is to detect duplicated pages and show one of them in search results. We found many similar pages getting crawled into repository. They are similar in content of the page but their URLs are different, which makes the crawler hard to detect them. Once the duplicated pages are stored in the repository, search engine will treat each page differently, and they can be shown in same result as duplicated lines. Therefore in our future work, we will find a technique to use to reduce this problem by either improve our crawler to be capable of detecting duplicate page from its content, or create a program to find and remove those pages before they being indexed.

In conclusion, we developed a prototype code search engine focusing on code snippets and text in tutorial pages. Our code search engine works well with free text queries and provides both texts and structural code snippets relevant to user queries. Therefore, developers can read and understand source code in a search result page, and text information about a code snippet may facilitate their understanding. They may be able to get necessary information without further clicking in a search result page. We hope that our code search engine will complement limitations of both general purpose search engines and code search engines currently available and better meet user needs on doing code search.

7. Additional data

A list of 33 seed tutorial websites:

(1) http://java.sun.com/docs/books/tutorial/
(2) http://learnola.com/
(3) http://www.zetcode.com/
(4) http://forum.codecall.net/java-tutorials
(5) http://www.dickbaldwin.com/java/
(6) http://www.learn-java-tutorial.com/
(7) http://www.developer.com/java/
(8) http://pages.cpsc.ucalgary.ca/~kremer/tutorials/Java/
(9) http://www.beginner-tutorials.com/java-tutorials.php
(10) http://www.javabeginner.com
(11) http://www.javacoffeebreak.com/
(12) http://www.cafeaulait.org/javatutorial.html
(13) http://www.javaworld.com/
(14) http://en.wikiversity.org/wiki/Java_Tutorial
(15) http://leepoint.net/notes-java/index.html
(16) http://www.javafaq.nu/java-example.html
(17) http://www.java-tips.org
(18) http://www.java2s.com/Tutorial/Java/CatalogJava.htm
(19) http://www.java2s.com/Code/Java/CatalogJava.htm
(20) http://www.java2s.com/Article/Java/CatalogJava.htm
(21) http://www.java2s.com/Code/JavaAPI/CatalogJavaAPI.htm
(22) http://www.java2s.com/Product/Java/GUI-Tools/CatalogGUI-Tools.htm
(23) http://www.tech-recipes.com/category/computer-programming/java-programming/
(24) http://www.exampledepot.com/egs/
(25) http://www.devdaily.com/java/
(26) http://www.roseindia.net/java/
(27) http://en.wikibooks.org/wiki/Java_Programming/
(28) http://www.codetoad.com/java/
(29) http://danzig.jct.ac.il/java_class/
(30) http://www.java-samples.com/showtitles.php?category=Java&start=1
(31) http://www.algolist.net/Algorithms/
(32) http://www.javapractices.com/
(33) http://home.cogeco.ca/~ve3ll/jatutor0.htm

8. References

[1] T. J. Biggerstaff, B. G. Mitbander, and D. Webster, "The concept assignment problem in program understanding," in *Proceedings of the 15th international conference on Software Engineering* Baltimore, Maryland, 1993, pp. 482-498.

[2] G. Fischer, S. Henninger, and D. Redmiles, "Cognitive tools for locating and comprehending software objects for reuse," in *Proceedings of the 13th International Conference on Software engineering* Austin, Texas: IEEE Computer Society Press, 1991, pp. 318-328.

[3] J. Li, S. Huffman, and A. Tokuda, "Good abandonment in mobile and PC internet search," in *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval* Boston, MA, USA, 2009, pp. 43-50.

[4] V. Rajlich and N. Wilde, "The role of concepts in program comprehension," in *Proceedings of the 10th International Workshop on Program Comprehension*, 2002, pp. 271-278.

[5] M. Umarji, S. Sim, and C. Lopes, "Archetypal Internet-Scale Source Code Searching," in *Open Source Development, Communities and Quality*, 2008, pp. 257-263.