Extracting knowledge from Java libraries

# Problem context

A large volume of useful Java libraries is available in the form of jar files containing compiled Java source code. These libraries range from stand-alone complete applications to small utilities or APIs for performing wide variety of tasks. Some examples are:

1. Google GSON: <https://mvnrepository.com/artifact/com.google.code.gson/gson>
2. Apache Commons Collections: <https://mvnrepository.com/artifact/commons-collections/commons-collections>
3. JSoup HTML parser: <https://mvnrepository.com/artifact/org.jsoup/jsoup>
4. Apache PDFBox: <https://mvnrepository.com/artifact/org.apache.pdfbox/pdfbox>

Needless to say, there are hundreds of useful and popular libraries available as open source.

We would like to analyse the information available (as retrieved via javap tool) in .class files to do the following:

1. Find out any significant patterns in the usage of different JVM instructions.
2. Build/train a ML model to detect type/functionality of software by looking at the binary file (i.e .class or .jar files)

# Possible approach

Our main source of raw data would be Maven repositories that contain categorised binaries for various types of software. For instance take a look at <https://mvnrepository.com/open-source>. It shows several categories of libraries such as:

|  |  |  |
| --- | --- | --- |
| [Aspect Oriented](https://mvnrepository.com/open-source/aop-programming)  [Actor Frameworks](https://mvnrepository.com/open-source/actor-frameworks)  [Application Metrics](https://mvnrepository.com/open-source/application-metrics)  [Build Tools](https://mvnrepository.com/open-source/build-tools)  [Bytecode Libraries](https://mvnrepository.com/open-source/bytecode-libraries) | [Command Line Parsers](https://mvnrepository.com/open-source/command-line-parsers)  [Cache Implementations](https://mvnrepository.com/open-source/cache-implementations)  [Cloud Computing](https://mvnrepository.com/open-source/cloud-computing-integration)  [Code Analyzers](https://mvnrepository.com/open-source/code-analyzers)  [Collections](https://mvnrepository.com/open-source/collections) | [Configuration Libraries](https://mvnrepository.com/open-source/config-libraries)  [Core Utilities](https://mvnrepository.com/open-source/core-utilities)  [Date and Time Utilities](https://mvnrepository.com/open-source/date-time-utilities)  [Dependency Injection](https://mvnrepository.com/open-source/dependency-injection)  [Embedded SQL Databases](https://mvnrepository.com/open-source/embedded-sql-databases) |

In order to achieve our goals, we can perform the following tasks:

1. Create a program to crawl (or fetch via API if available from mvnrepository.com) the information about various categories of libraries. The information to be fetched will include:
   1. Category name
   2. Category detailed description if any
   3. Number of libraries available in each category
2. For each library found in a category fetch the following:
   1. All the metadata available. E.g. License, Categories, Tags, Used By, Number of versions available and usage of each version etc.
   2. Download the jar file(s) for the latest as well as the most used version of the library.
3. All of the above information should be stored in a database table(s).
4. Write a program to process the downloaded jar files as follows:
   1. Find fully qualified names (FQN) for each of the class available in a jar file
   2. For each FQN invoke javap tool to find the internal information about the class. You will have to parse the output of javap to extract the desired information which will include:
      1. Count of occurence of different JVM instructions
      2. Number of methods
      3. Constant pool size
      4. Count of different classes used
      5. And so on
   3. Save this information in a database table(s). Records should be properly linked (via FKs) to metadata related table(s).
5. Write another program which can make use of the above extracted information to train a ML model such that following types of labels may be assigned to each training sample:
   1. Categories of the software to which a training sample belongs.
   2. Tags assigned to the library.
   3. Usage levels
   4. And so on

# Skills you need to have/acquire to complete this task

1. Programming in a high-level language preferably Java or Python.
2. Creating database tables to model the entities and their relationships and working with SQL.
3. Using one of the ML libraries. E.g. in Python you have [scikit-learn](http://scikit-learn.org/stable/), and in Java you may use libraries such as [Apache MLlib](https://spark.apache.org/docs/latest/ml-guide.html), [JavaML](http://java-ml.sourceforge.net/), [Deeplearning4j](https://deeplearning4j.org/index.html) etc. I think it is not difficult to use these libraries if you are comfortable writing non-trivial programs in Python or Java.