Goal: Compute the ability of classes to fit into code completion.
Author: Martin Monperrus

In this exercise, you will study the ability of classes to fit into code completion.

Preparation

Download dataset.zip on Moodle. This file contains the extraction of type usages for 30 Java classes on 10000 Jar files. One line represents a type-usae, i.e. a variable of a given type and the method calls observed statically on this variable. For instance, for StringBuffer "call:append(String) call:toString()" encodes the fact that in a code base a variable of type StringBuffer has two methods called on it. Choose one class among the 30 files and post its name to the forum.

Write a small parser to parse the type-usage file and encode this data structure into memory.

Analysis

You will now write a small code completion simulator, it will produce the trace given in appendix. The structure of the written report is also given in appendix.

What is the number of different methods of this class? Describe in a few sentences the top-5 most used methods.

Code Completion Simulation #1

A first code completion simulator consists of studying the number of proposals per prefix size (for instance: "a" -> 10 methods, "b" ->2 methods, etc.). Compute the number of prefixes per size and the average and median number of methods per prefix size.
For instance:
prefixes of size 1: (3 prefixes, average 6 methods, median 10 methods)
  a -> 10 methods
  s -> 13 methods
  r -> 1 methods:
prefixes of size 2: (5 prefixes, average: 4.8 methods, median 4 methods)
  ab -> 6 methods
  af -> 4 methods
  se -> 12 methods
  se -> 1 methods
  re -> 1 methods:
etc.
For this class, what are the main factors behind the evolution of those metrics?

Code Completion Simulation #2

A second code completion simulator consists of removing random calls and see whether a nearest-neighbor approach can find them again, as follows (k as parameter of the algorithm):

For each type-usage (line of the file):
  remove 50% of the calls (rounded to upper value)
  compute the nearest neighbors (type-usages with up to k additional method calls)
  predict the method calls that appear in the nearest neighbors but not in this type usage
  compute precision and recall
Return: average and median precision and recall

Eventually, we obtain precision and recall for different values of k:

```
<result k="1">
<precision average=".8" median=".7"/>
<recall average=".5" median=".9"/>
</result>
<result k="2">
<precision average=".7" median=".6"/>
<recall average=".6" median=".95"/>
</result>
....
```

Some type-usages are bad with respect to code completion (low precision and recall). Give an explanation for two of them.

Appendix

Command line interface
$ java fr.univ-lille1.iagl.CodeCompletionSimulator java.util.List.dat

Trace:
$ java

<analysis class="java.util.List">
<nb-method>76</nb-method>
<simulation1>
<data1>
<x prefix-size="1" number-prefixes="32"/>
<x prefix-size="2" number-prefixes="11"/>
...
</data1>
<data2>
<x prefix="a" value="10"/>
<x prefix="c" value="5"/>
..
<x prefix="set" value="7"/>
...
</data3>
<plot>
<point prefix-size="1" average-number="12" median-number="9"/>
<point prefix-size="2" average-number="7" median-number="4"/>
...
</plot>
</simulation1>
<simulation2>
<number-of-type-usages>61536</number-of-type-usages>
<plot><!-- from k=1 to k=10 -->
<result k="1">
<precision average=".8" median=".7"/>
<recall average=".5" median=".9"/>
</result>
....
</plot>
</simulation2>
</analysis>

Written Report Structure

Presentation of the class (1 paragraph)
Top-5 most used methods
Simulation #1
  Plot1 + explanatory paragraph / Plot2 + explanatory paragraph
Simulation #2
  Plot + Explanation of type-usages not good for code completion
Appendix
  Trace of program execution (also submitted to Moodle directly)