

Software Quality Measurement

Gruppe H

5. Januar 2018

Inhaltsverzeichnis

1	Software Quality Measurement	3
1.1	CBO - Kopplung zwischen Objektklassen	3
1.2	LCOM - Mangel an Abgeschlossenheit	4
1.3	Weitere Kenngrößen und graphische Darstellung	5
1.4	Vergleich mit einer abgeänderten Version von Mallet	8

1 Software Quality Measurement

Für `mallet`¹, eine Verarbeitungssoftware für natürliche Sprache, sollen beispielhaft Kenngrößen für Softwarequalität ermittelt und ausgewertet werden. Zur Berechnung der Kenngrößen wird `ckjm`² verwendet, das alle sechs Kenngrößen (Metriken zum Messen von Softwarequalität) nach Chidamber und Kemerer sowie zwei zusätzliche ermittelt. Die Ausgaben von `ckjm` werden mit Hilfe eines zur Verfügung gestellten, auf `matplotlib` basierenden Pythonskripts aufbereitet und in Form von Histogrammen dargestellt.

Listing 1.1: Beispiele für den Output von `mallet`

```
1 cc.mallet.extract.test.TestDocumentExtraction 7 0 0 20 31 21
   0 7
2 cc.mallet.pipe.SerialPipes$Predicate 2 1 0 2 3 1 1 2
3 cc.mallet.classify.BalancedWinnowTrainer 6 0 0 11 21 11 1 6
4 cc.mallet.util.CharSequenceLexer 20 1 0 1 45 94 14 16
5 ...
```

1.1 CBO - Kopplung zwischen Objektklassen

Ein hoher Wert des CBO (Coupling between object classes) ist erwünscht. Das bedeutet, dass Methoden mit vielen Instanzvariablen eng gekoppelt sind, was sich wiederum positiv auf die Softwarequalität auswirkt.

Ein kleiner Wert ist unerwünscht, da dies bedeutet, dass alle Methoden der Klasse eigene unabhängige Instanzvariablen benutzen. Dies wirkt sich negativ auf die Softwarequalität aus. Über 10% der Klassen haben einen CBO von 0, die Tabelle der schlechtesten Klassen zeigt also nur eine Teilmenge der Klassen mit diesem Wert.

CBO: schlechteste Klassen	
<code>cc.mallet.util.Lexer</code>	0
<code>cc.mallet.topics.TopicReports</code>	0
<code>cc.mallet.cluster.util.PairwiseMatrix</code>	0

CBO: beste Klassen	
<code>cc.mallet.classify.tests.TestNaiveBayes</code>	33
<code>cc.mallet.fst.semi_supervised.tui.SimpleTaggerWithConstraints</code>	34
<code>cc.mallet.fst.tests.TestCRF</code>	51

¹Mallet af github

²ckjm auf github, Manual

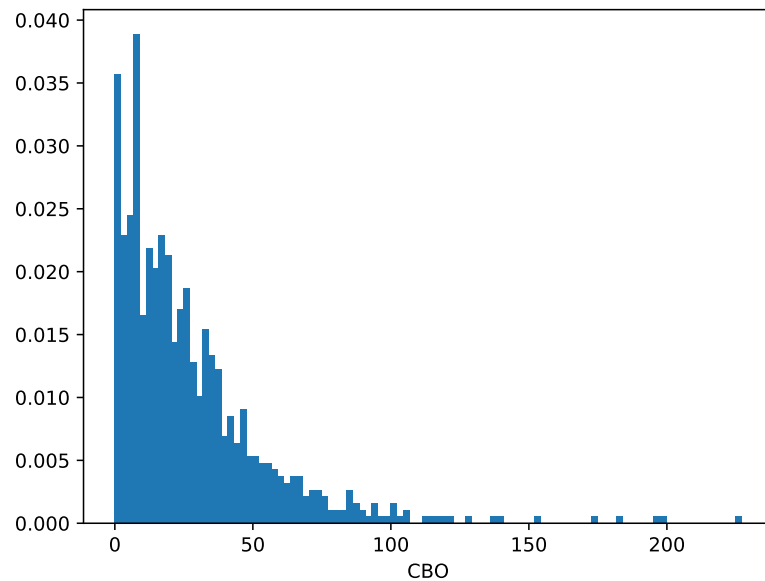


Abbildung 1.1: CBO: Coupling between Object Classes

Das Histogramm (Abb. 1.1) zeigt die CBO-Werte aller getesteten Klassen. Die CBO-Werte liegen im Bereich von 0 bis 51.

1.2 LCOM - Mangel an Abgeschlossenheit

Im Vergleich zum CBO-Wert verhält sich der LCOM-Wert komplementär. Ein hoher Wert steht für den Mangel an Kohäsion (Zusammenhang zwischen den Methoden) und wirkt sich somit negativ auf die Softwarequalität aus. Ein niedriger Wert bedeutet keinen Mangel und ist somit positiv für die Softwarequalität.

LCOM: beste Klassen

<code>cc.mallet.fst.semi_supervised.GELatticeTask</code>	0
<code>cc.mallet.optimize.OptimizerEvaluator\$ByBatchGradient</code>	0
<code>cc.mallet.util.ProgressMessageLogFormatter</code>	0

LCOM: schlechteste Klassen

<code>cc.mallet.types.InstanceList</code>	2365
<code>cc.mallet.types.MatrixOps</code>	1911
<code>cc.mallet.fst.CRF</code>	1578

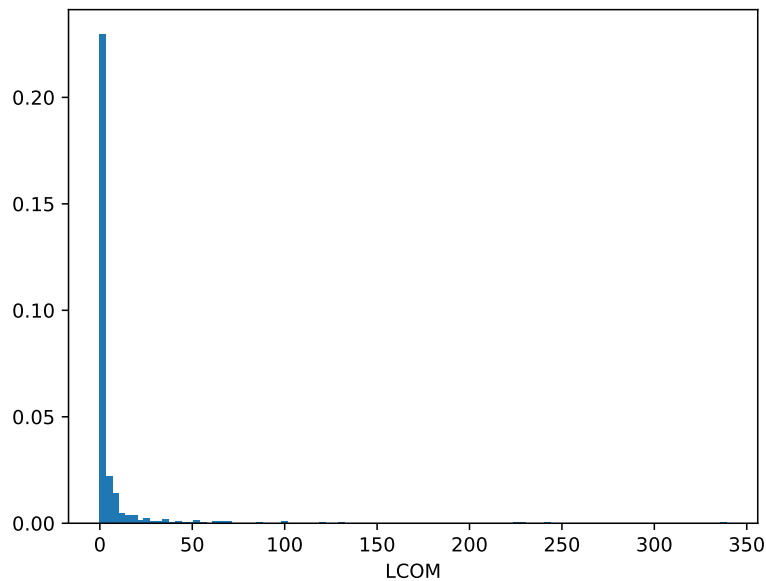


Abbildung 1.2: LCOM: Lack of cohesion in methods

1.3 Weitere Kenngrößen und graphische Darstellung

Die Kenngrößen werden als normierte Histogramme dargestellt, auf der x-Achse sind Bereiche möglicher Werte der Kenngröße aufgetragen. Auf der y-Achse der Anteil der Klassen, die diesem Wertebereich zugeordnet sind.

Neben den in den vorherigen Kapiteln bereits vorgestellten Kenngrößen wurden folgende weitere von **ckjm** ermittelt:

WMC: Weighted methods per class (Abb. 1.3 li.) stellt der Wert einfach nur die Anzahl der Methoden einer Klasse dar, weil die Gewichtung (entsprechend der Komplexität) aller Methoden auf 1 gesetzt wurde. Als Grund für diesen festen Wert werden die einfache Implementierung und die Kompatibilität zu Chidamber und Kemerer genannt. Für die Verständlichkeit und Erweiterbarkeit ist es von Vorteil, wenn die Anzahl der Methoden einer Klasse in einem überschaubaren Rahmen bleiben. Bei der betrachteten Software haben alle Klassen 0–5 Methoden, was sehr überschaubar ist.

- WMC betrachtet die Komplexität (Erweiterbarkeit und Verständlichkeit) der Klassen. Größere Werte sind negativ. Kleinere Werte sind positiv.

DIT: Depth of Inheritance Tree (Abb. 1.3 re.)

- DIT misst die Tiefe des Vererbungsbaums. Je tiefer dieser Baum ist, desto negativer wirkt sich das auf die Softwarequalität aus, weil es dann schwie-

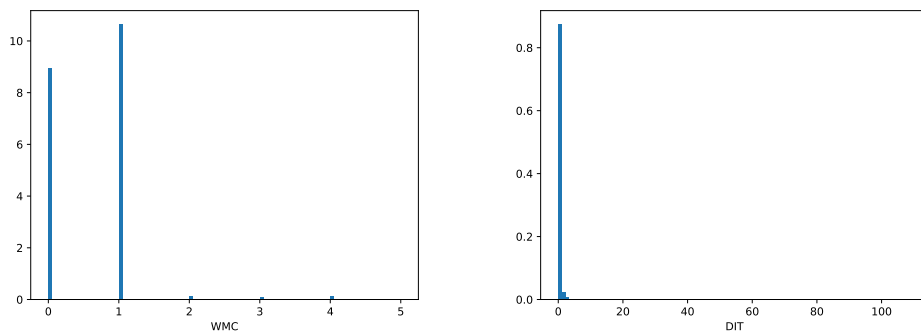


Abbildung 1.3: WMC (WB: 0-5) und DIT (WB: 0-109)

riger ist, die Vererbungsstruktur bis in die Blätter nachzuvollziehen. D.h. in unserem Histogramm sind solche Klassen gut, die den Wert 0 haben, weil die Tiefe des Baums

NOC: Number of Children (Abb. 1.4 li.)

- NOC misst die Anzahl der Subklassen, die direkt unter der gemessenen Klasse existieren. Ein hoher Wert bedeutet, dass diese Klasse oft wiederverwendet wird und somit Code gespart wird. Das bedeutet aber auch, dass eine Klasse mit hohem Wert gut getestet muss, um die Softwarequalität zu steigern. Im Vergleich zu DIT misst NOC die Breite des Baums, statt der Tiefe.

RFC: Response for a Class (Abb. 1.4 re.)

- Der RFC Wert steht für die Anzahl aller möglichen auszuführenden Methoden innerhalb der Klasse. Je höher der Wert, desto höher die Komplexität der Klasse. Also ist ein hoher Wert nicht erwünscht.

Ca: Afferent coupling (not a C&K metric, Abb. 1.5 li.) gibt an, wie viele Klassen von dieser Klasse abhängen. Eine Klasse mit einem hohen Ca-Wert hat eine hohe Verantwortung innerhalb der Software, weil sie Funktionalitäten für viele andere Klassen zur Verfügung stellt.

- Der Ca Wert bestimmt die Anzahl der Packages, die von der Package dieser Klasse abhängen. Ein hoher Wert bedeutet eine hohe Verantwortung für diese Package und ist somit negativ behaftet. Da unsere Klassen größtenteils niedrigere Werte besitzen, ist dies positiv für die Softwarequalität.

NPM: Number of Public Methods for a class (not a C&K metric) (Abb. 1.5 re.)

- Der NPM Wert bestimmt die Anzahl der public methods innerhalb der Klasse. Ein hoher Wert bedeutet eine große Verantwortung und hohe Komplexität und ist daher für die Softwarequalität nicht von Vorteil.

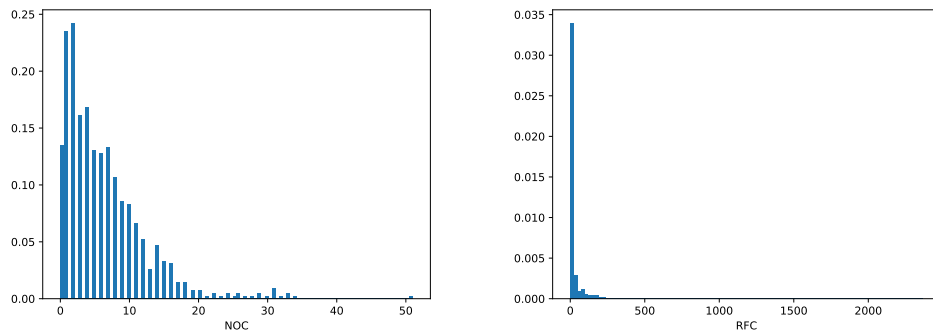


Abbildung 1.4: NOC (WB: 0–51) und RFC (WB: 0–2356)

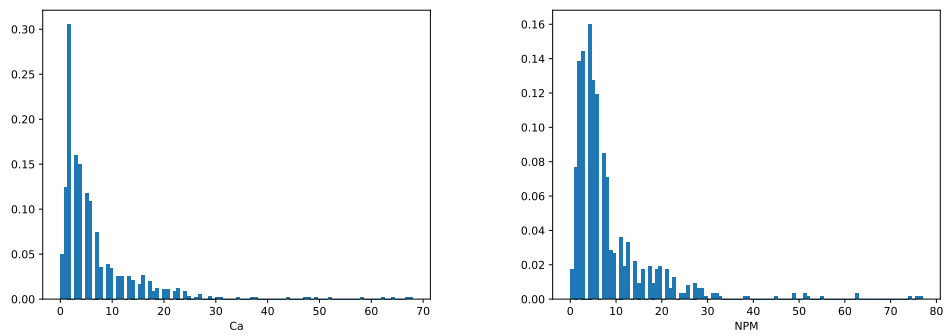


Abbildung 1.5: Ca (WB: 0-68) und NPM (0-77)

1.4 Vergleich mit einer abgeänderten Version von `mallet`

Dem Programm `mallet` werden zwei weitere Klassen `NewParallelTopicModel.java` und `TopicInferencerInterface.java` hinzugefügt. Nach dem Kompilieren wird die Analyse von `mallet` wiederholt und es werden die Kennzahlen der Klassen `NewParallelTopicModel` und `ParallelTopicModel` verglichen:

Klasse	WMC	DIT	NOC	CBO	RFC	LCOM	Ce	NPM
<code>NewParallelTopicModel</code>	68	1	0	22	233	800	0	61
<code>ParallelTopicModel</code>	63	1	2	21	227	627	11	58