Document Classification System

Test Report

2014/15 Semester A

Delivery Data: 5th Dec 2014

CS3343 LA1 – Acumen

Member Information

Role	Name	SID
PM	WANG Mingyang	52640166
Assistant PM	PU Jie	52640234
CM	FANG Zhou	52639099
Developer	FENG Xikang	52639689
Developer	WANG Yiji	52639040
Developer	ZHU Renjie	52639014

Table of Contents

1. Introduction

In order to make sure that our program can get correct documentation classification result for one specific input file, we have written nine test classes to test our program. In release 2, we just have 3 test cases because the source code of program hasn't finished in that time. After we finished our source code, we built nine test classes that each is corresponding to one class in our source code. Using this test classes, we refactor our source code and achieve a relatively accurate classification. Finally, the total coverage percentage of our test reaches 90.1%.

2. Test Classes

Test Class	Test Attributes/Cases	Purpose
ReadContentTestcase.java	processDocumentTestForSingleFile()	To test whether this class can read single file correctly
	processDocumentTestForFiles()	To test whether this class can read all files of one given folder
	processDocumentTestForSplit()	To test whether this class can filter out the useless characters, like @,#,$,%...
SingleFileWordCountTestcase.java	getCategoryTest()	To test whether this class can get correct category for one specific file or folder
	getCountTableTest()	To test whether this class can count correct total number of all words in given file
	getWordsNumTest()	To test whether this class can count correct total words number in given file
	getWordFrequencyTest()	To test whether this class can get correct word frequency of all words
	write2FileTest()	To test whether this class can write output content to file correctly
BuildDictionaryTestcase.java	common_word	To test whether this class can get the correct common words which appears in every given file
	total_word	To test whether this class can get correct word count for all words except common words in given files
BuildWordCountMatrixTestcase.java	categoryMap	To test whether this class can get correct category map
	fileNameMap	To test whether this class can get correct file name map
	wordCountSet	To test whether this class can get correct word count set
	fileNumInCategory	To test whether this class can get correct total file number in one category
CalculateIGTestcase.java	categoryNum	To test whether this class can get correct total category in given folder
	totFileNum	To test whether this class can

		get correct total file number of given folder
	num_file_in_one_cat	To test whether this class can get correct file number for one specific category
	matrix	To test whether this class can build correct words frequency matrix
	info_gain	To test whether this class can calculate correct information gain for each word in given files
FeatureSelectionTestcase.java	feature	To test whether this class can get correct feature words of all categories
BuildDocVectorTestcase.java	dv	To test whether this class can get correct document vector before normalization
	matrix	To test whether this class can get correct document matrix which store each total number of each word for each file
	wordNumInFileList	To test whether this class can get correct total word number in each file
BuildVectorForInputFileTestcase.java	dv	To test whether this class can get correct document matrix before normalization for input file
	matirx	To test whether this class can get correct document matrix which store each total number of each word for each file
CalculateSimilarityTestcase.java	inDocVector	To test whether this class can product correct input document vector
	vectorMatrix	Test whether this class can get correct vector matrix
	similarity	Test whether this class can calculate correct similarity value between input file and each files in database
	catMap	To test whether this class can get correct category map
	startIndexOfCat	To test whether this class can get correct start index for each category

	cosTest()	To test whether this class can calculate correct cos value
Classifier.java	MainTest1()	To test whether this class can print correct warning information when no parameter is given
	MainTest2()	To test whether this class can get correct classification for one given file

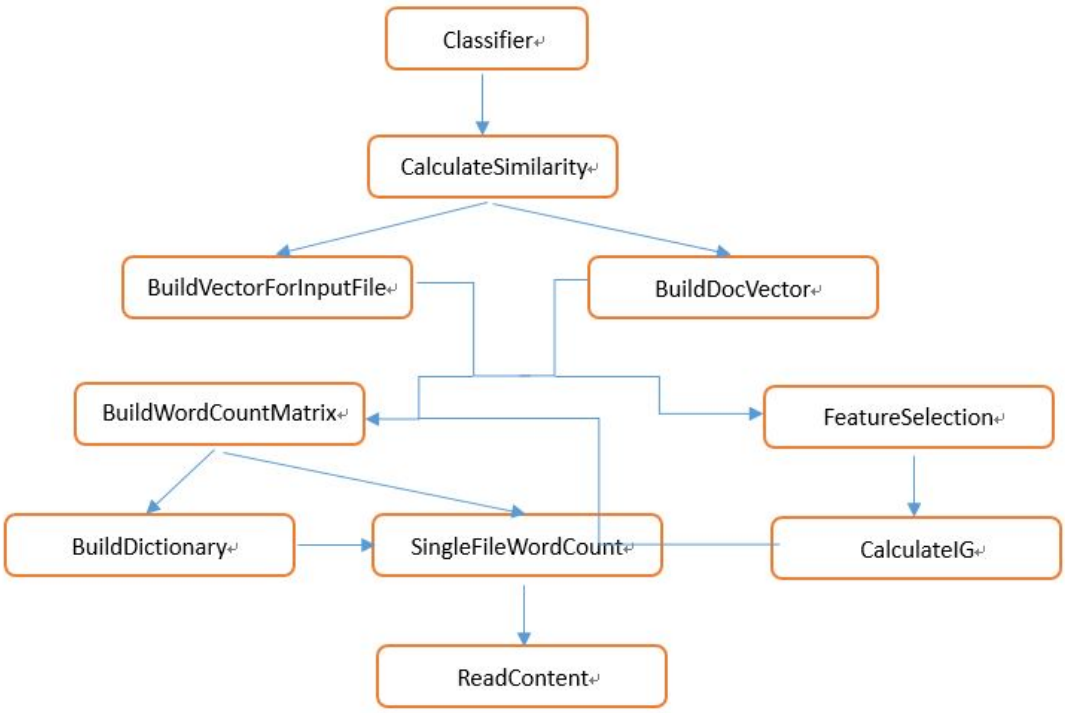
3. Test Procedure

3.1 Test objectives

In order to get correct documentation classification result for one given input file, we must make sure that the output of each internal step is correct. Therefore, our group adopts Bottom-up integration testing approach to achieve this goal.

3.2 Test Design

Test Approach: Bottom-up Integration Testing Approach
Hierarchy Graph of Our Program Modules



3.3 Test Sequence

3.3.1 Unit Testing

Unit Testing 1: ReadContent (Test stub: N/A)

3.3.2 Integration Testing

Integration Testing 1:
SingleFileWordCount + ReadContent (Stub:N/A)

Integration Testing 2:
BuildDictionary + SingleFileWordCount + ReadContent (Stub: N/A)

Integration Testing 3:
BuildWordCountMatrix+BuildDictionary+ SingleFileWordCount + ReadContent (Stub: N/A)

Integration Testing 4:
CalculateIG + BuildWordCountMatrix+BuildDictionary+ SingleFileWordCount + ReadContent (Stub: N/A)

Integration Testing 5:
FeatureSelection + CalculateIG + BuildWordCountMatrix+BuildDictionary+ SingleFileWordCount + ReadContent (Stub: N/A)

Integration Testing 6:
BuildDocVector + FeatureSelection + CalculateIG + BuildWordCountMatrix+BuildDictionary+ SingleFileWordCount + ReadContent (Stub: N/A)

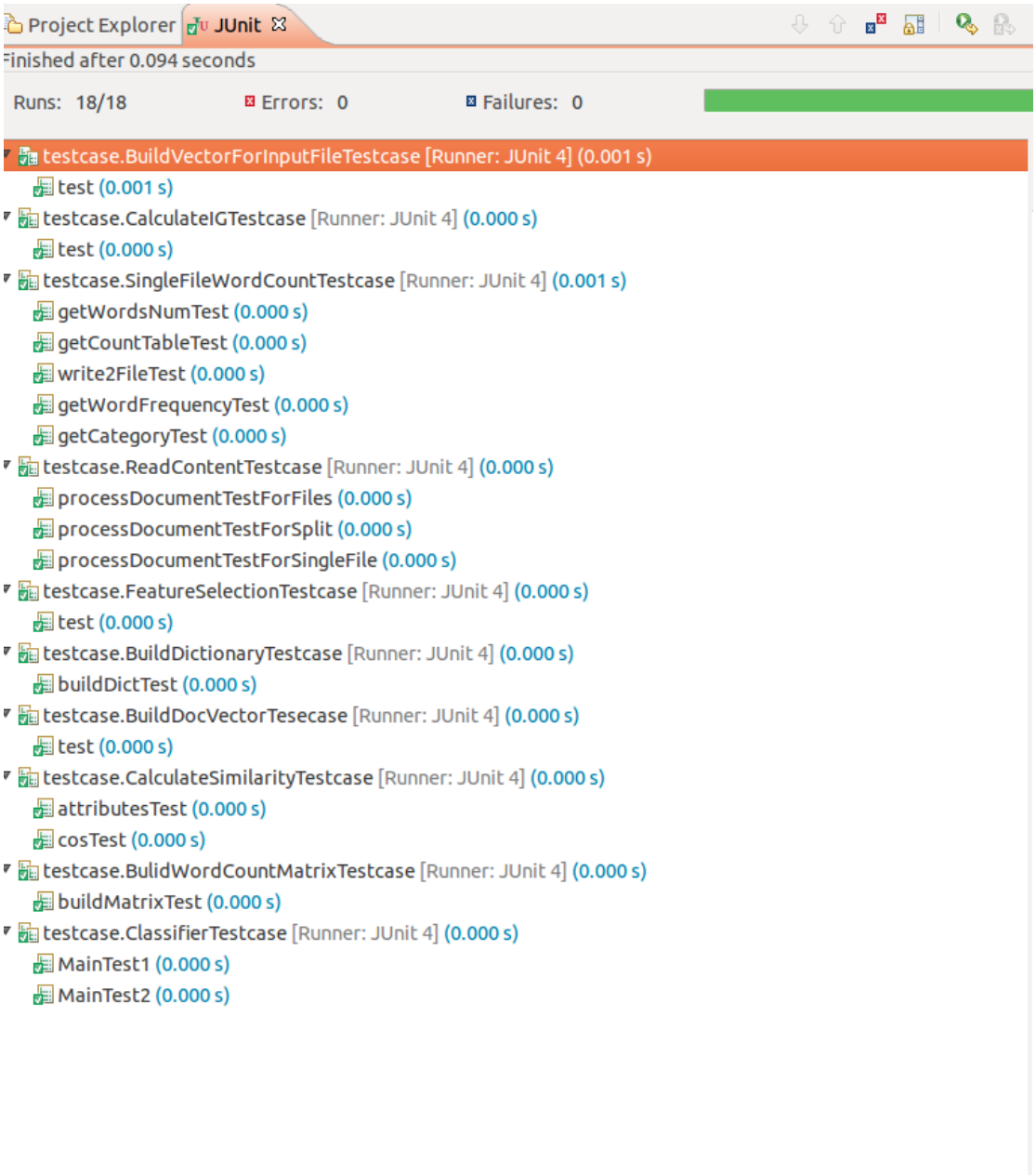
Integration Testing 7:
BuildVectorForInputFile + FeatureSelection + CalculateIG + BuildWordCountMatrix+BuildDictionary+ SingleFileWordCount + ReadContent (Stub: N/A)

Integration Testing 8:
CalculateSimilarity + BuildVectorForInputFile + BuildDocVector + FeatureSelection + CalculateIG + BuildWordCountMatrix+BuildDictionary+ SingleFileWordCount + ReadContent (Stub: N/A)

3.3.3 System Testing

Classifier + CalculateSimilarity + BuildVectorForInputFile + BuildDocVector + FeatureSelection + CalculateIG + BuildWordCountMatrix+BuildDictionary+ SingleFileWordCount + ReadContent (Stub: N/A)

4. Test Execution



5. Test Coverage Result

Coverage

Session: Acumen (Nov 30, 2014 9:52:55 PM)

Counter	Coverage	Covered	Missed	Total
Instructions	95.0 %	6,999	366	7,365
Branches	85.3 %	186	32	218
Lines	92.9 %	871	67	938
Methods	100.0 %	85	0	85
Types	100.0 %	22	0	22
Complexity	86.6 %	168	26	194

Acumen (Nov 30, 2014 9:52:55 PM)

Element	Coverage	Covered Instructions	Missed Instructions
Acumen	95.0 %	6,999	366
src	95.0 %	6,999	366
classifier	90.1 %	2,730	299
ReadContent.java	92.2 %	118	10
ReadContent	92.2 %	118	10
ReadContent(String)	100.0 %	11	0
arrayCopy(String[])	100.0 %	31	0
readAndFilter(File)	80.6 %	29	7
processDocument()	94.0 %	47	3
SingleFileWordCount.java	97.9 %	141	3
SingleFileWordCount	97.9 %	141	3
getCategory()	100.0 %	3	0
getCountTable()	100.0 %	3	0
getWordsNum()	100.0 %	3	0
SingleFileWordCount(String, St	100.0 %	19	0
write2File(String)	94.7 %	54	3
getWordFrequency()	100.0 %	59	0
FeatureSelection.java	95.1 %	155	8
FeatureSelection	94.6 %	141	8
featureSelect(int, String, String)	96.0 %	24	1
writeFeatureFile(String)	94.3 %	50	3
sortMap(String)	93.7 %	59	4
BuildDictionary.java	96.2 %	205	8
BuildDictionary	96.2 %	205	8
writeToFile(String, String, Hash	95.6 %	87	4
buildDict(String, String, ArrayLi	96.6 %	115	4
BuildVectorForInputFile.java	97.3 %	248	7
BuildVectorForInputFile	97.3 %	248	7
BuildVectorForInputFile(String,	100.0 %	32	0
write2File()	93.0 %	40	3
initFeatures()	95.3 %	82	4
buildVector()	100.0 %	94	0
BuildDocVector.java	98.1 %	359	7
BuildDocVector	98.1 %	359	7
BuildDocVector(String, String, S	100.0 %	37	0
normalizeVector()	95.8 %	68	3
buildVector()	100.0 %	116	0
initFeatures()	97.2 %	138	4

Classifier.java	39.2 %	149	231
Classifier	39.2 %	149	231
main(String[])	38.7 %	146	231
CalculateIG.java	98.0 %	395	8
CalculateIG	98.0 %	395	8
CalculateIG(String, String, String	100.0 %	25	0
write2File()	94.8 %	55	3
initMatrix(String, String)	96.0 %	120	5
getIG()	100.0 %	195	0
CalculateSimilarity.java	98.0 %	395	8
CalculateSimilarity	97.9 %	378	8
categoryBelongsTo(int)	90.0 %	36	4
cos(ArrayList<Double>, ArrayLi	100.0 %	56	0
classify()	100.0 %	109	0
new Comparator() {...}	100.0 %	17	0
CalculateSimilarity(String, Strin	97.8 %	177	4
BuildWordCountMatrix.java	98.4 %	565	9
BuildWordCountMatrix	98.4 %	565	9
BuildWordCountMatrix(String,	100.0 %	29	0
printList(ArrayList<Integer>, St	93.9 %	46	3
printMatrix(HashMap<String, A	96.1 %	74	3
buildWordMatrix()	100.0 %	195	0
buildMatrix()	98.7 %	221	3
testcase	98.5 %	4,269	67
testcase	98.5 %	4,269	67
ClassifierTestcase.java	66.7 %	52	26
ClassifierTestcase	66.7 %	52	26
setUp()	100.0 %	3	0
getOutput()	100.0 %	7	0
MainTest1()	80.0 %	12	3
setOutput()	100.0 %	15	0
MainTest2()	23.3 %	7	23
FeatureSelectionTestcase.java	95.2 %	100	5
FeatureSelectionTestcase	95.2 %	100	5
setUp()	100.0 %	48	0
test()	89.8 %	44	5
BuildDictionaryTestcase.java	100.0 %	131	0
BuildDictionaryTestcase	100.0 %	131	0
setUp()	100.0 %	46	0
buildDictTest()	100.0 %	72	0
SingleFileWordCountTestcase.java	94.8 %	184	10
SingleFileWordCountTestcase	94.8 %	184	10
getCategoryTest()	100.0 %	8	0
getCountTableTest()	100.0 %	9	0
getWordsNumTest()	100.0 %	9	0
getWordFrequencyTest()	83.3 %	25	5
setUp()	100.0 %	36	0
write2FileTest()	94.2 %	81	5
ReadContentTestcase.java	100.0 %	203	0
ReadContentTestcase	100.0 %	203	0
setUp()	100.0 %	1	0
processDocumentTestForSingle	100.0 %	35	0
processDocumentTestForSplit(	100.0 %	63	0
processDocumentTestForFiles(	100.0 %	83	0
BulidWordCountMatrixTestcase.ja	98.2 %	278	5
BulidWordCountMatrixTestcase	98.2 %	278	5
buildMatrixTest()	94.2 %	81	5
setUp()	100.0 %	167	0
CalculateSimilarityTestcase.java	99.0 %	615	6
CalculateSimilarityTestcase	99.0 %	615	6
cosTest()	96.3 %	79	3
attributesTest()	96.5 %	82	3
setUp()	100.0 %	390	0
BuildVectorForInputFileTestcase.ja	99.3 %	756	5
BuildVectorForInputFileTestcase	99.3 %	756	5
test()	86.8 %	33	5
setUp()	100.0 %	701	0
CalculateIGTestcase.java	99.5 %	905	5
CalculateIGTestcase	99.5 %	905	5
test()	93.9 %	77	5
setUp()	100.0 %	802	0
BuildDocVectorTesecase.java	99.5 %	1,045	5
BuildDocVectorTesecase	99.5 %	1,045	5
test()	90.4 %	47	5
setUp()	100.0 %	971	0