Document Classification System

Object Orientation Analysis and Design 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 Content

1. Introduction

In this information age, increasing amount of information is conveyed in digital form. Our system aims to act as a smart "librarian" who can tell the catalogue of the document belongs to within the binary codes. The process of classifying documents to catalogues is no longer painstaking and time-consuming with help of this system. User can immediately get the catalogue of the document by simply input the location of the document, getting rid of the need to read the document from the beginning. Designed to be "smart", the system will "learn" after each task it performs by expanding its own word library, making itself more and more powerful.

2. User Story

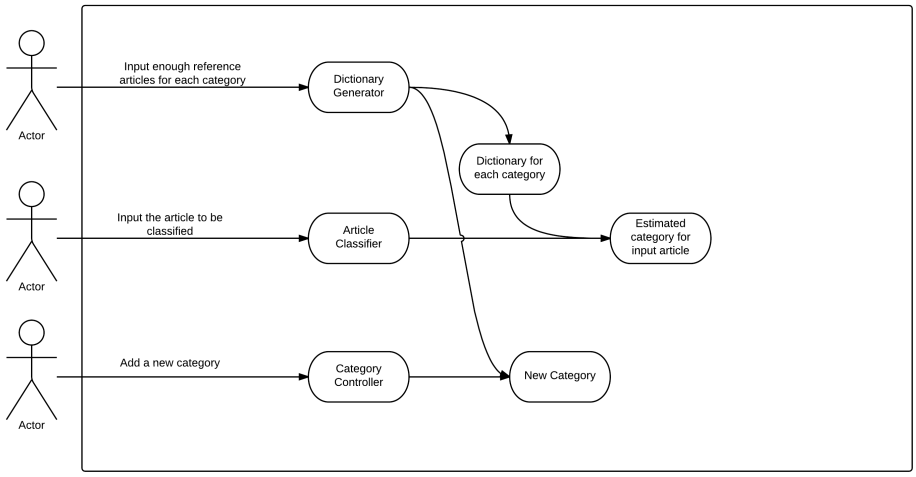
#	Name	Description
US1	Input training resource	**Input enough training resource and start the tool.** User can provide his/her own resources for tool training or just use the ones pre-located by us in the folder. Once the tool is started, a dictionary will be generated as criteria for classifying the document.
US2	Input document path	**Input the path of the document to be classified.** User needs to input the path of the document and type "enter" to proceed on the classification.
US3	Add catalogue	**Input new resource for new catalogue and re-start the tool.** To make the tool even smarter, user can provide more training resources to the system. By re-starting the tool, the system will update information in the word dictionary as well as other configurations.

3. Combined User Story

Mr. X has thousands of documents stored in his disk drive. Some are well classified into different folders of catalogue while others are in a mess with arbitrary location. Now Mr. X wants to know what catalogue a specific document belongs to but he does not have enough time to read the entire document with 100 pages nor could he get any information from its table of contents. So Mr. X places some well-classified documents under the root folder of the tool and start the tool. By simply inputting the path of the specific document and waiting for few seconds, the catalogue of the document is display by the system. Mr. X thus saves a lot of time for reading. Next time for a new document, he just needs to input the path and he will reach the result fast.

4. Use Case

4.1 Use Case Diagram



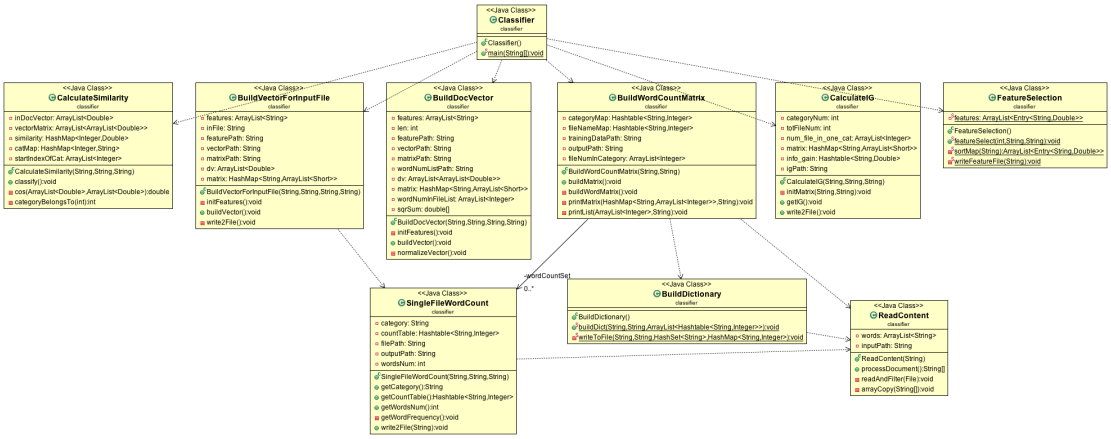
4.2 Use Case Table

User responsibility	**System responsibility**
1. User provides enough reference resources for each catalogue	
2. User starts the tool	
	3. System builds the dictionary with given input
4. User inputs the path of the document needed to be classified	
	5. System analyzes the document according to the dictionary generated
	6. System outputs the result catalogue of the document

5. Domain Modeling

5.1 UML Diagram

(Please view appendix I for full chart)

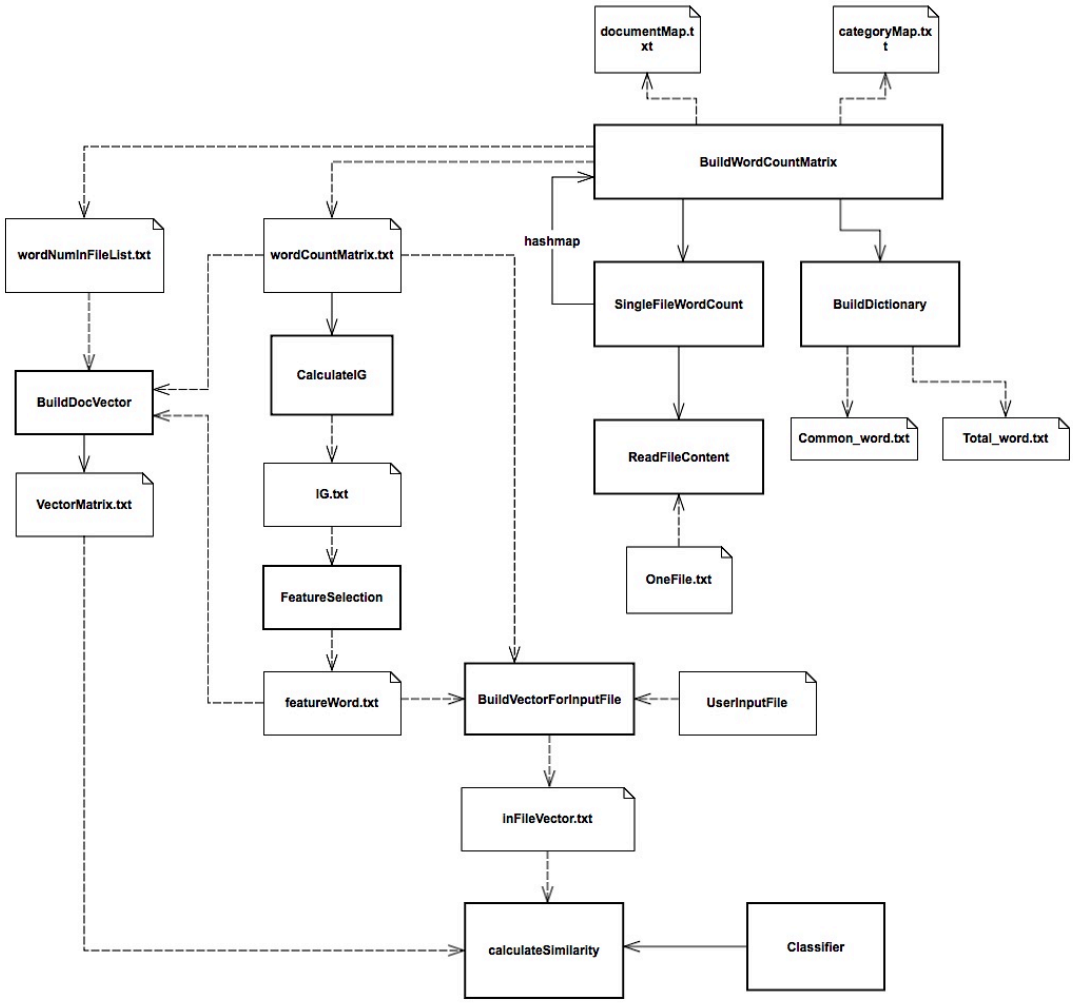


Classes:

BuildDictionary, BuildDocVector, BuildVectorForInputFile, BuildWordCountMatrix, CalculateIG, CalculateSimilarity, Classifier, FeatureSelection, ReadContent, SingleFileWordCount

5.2 Domain Modeling

5.2.1 Domain Modeling Diagram



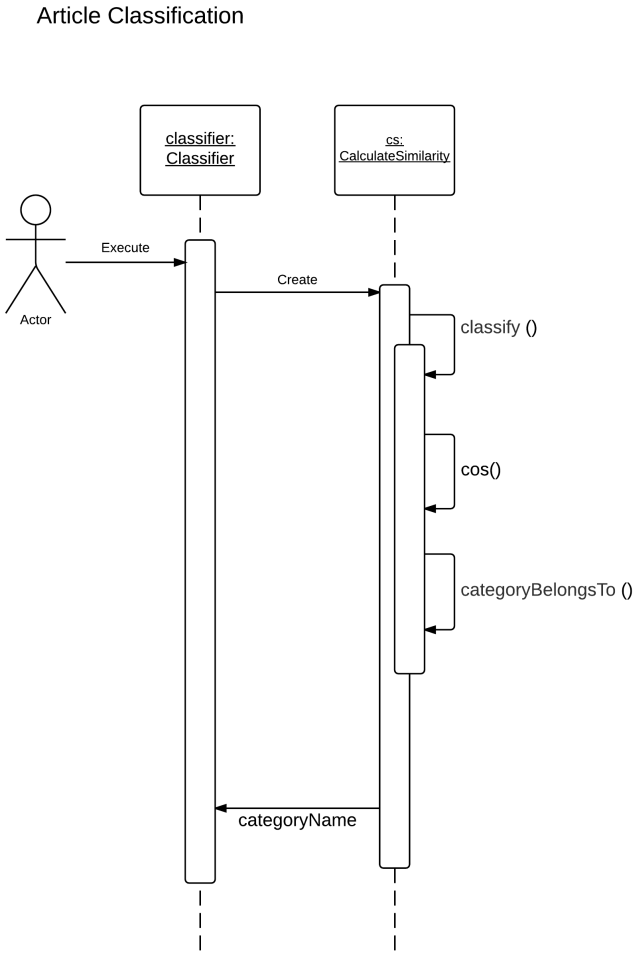
Structure diagram	Function
Rectangle	Class
Rectangle with a triangle on the right upper corner	Document
Dash line	Input/Output document
Solid line	Call classes/return value

5.2.2 Domain Entity Table

Domain Entity	**Responsibility**
BuildWordMatrix	Build word matrix file for storing data
SingleFileWordCount	Get counts for all words in one file
BuildDictionary	Build dictionary for common word and total word (word and their count)
ReadFileContent	Read all contents from a file
BuildDocVector	Build vectors for all feature words
Calculate IG	Calculate the information gain for each word in the dictionary
FeatureSelection	Select the feature words according to the information gain of each word
BuildVectorForInputFile	Build vectors for the words from user input file
calculateSimilarity	Compare the user input file vectors with the vector matrix built in the initialization stage
Classifier	Do file classification, return the category the file belongs to

5.3 Sequence Diagram

5.3.1 Article Classification

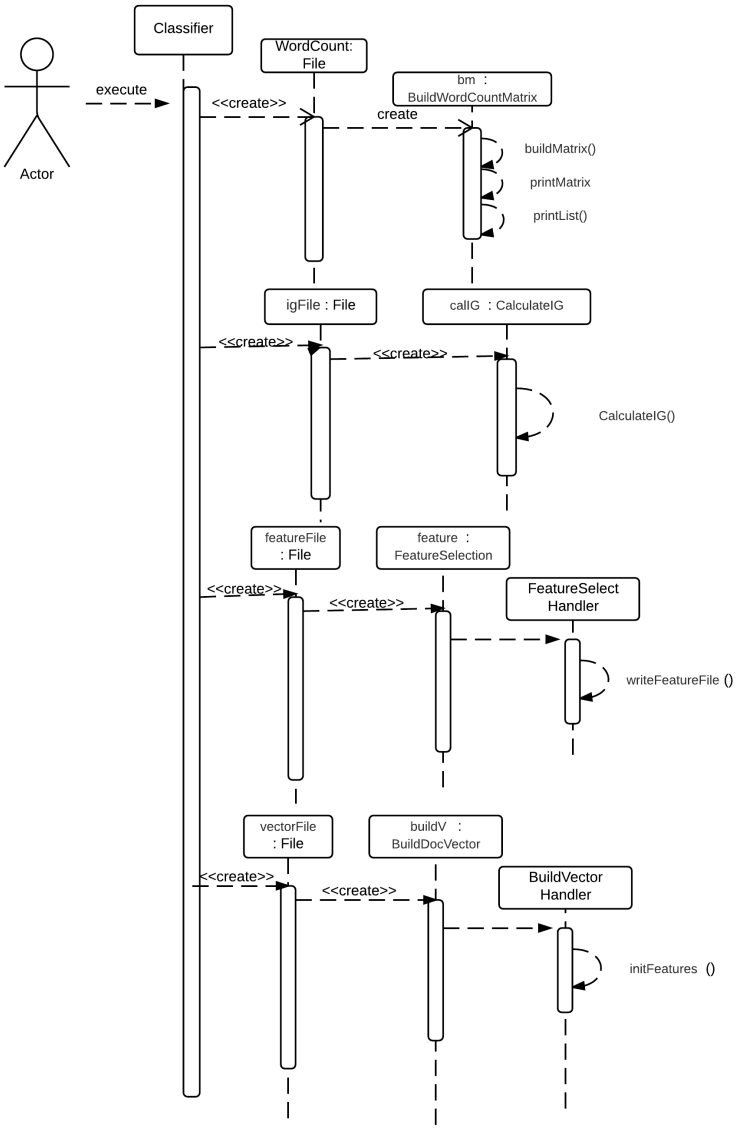


Function:

Get input file, analyze the file and return with the category the file belongs to.

5.3.2 Vector Generation

Build Vector For Feature Words



Function:

Build word dictionary (word matrix), filter the common words and return featured words that may define the category of the file.

6. Summary

In this Analysis report, our design fulfills users' requirements perfectly.

Firstly, users are able to set the desired categories. By providing sufficient number of articles, a list of featured words will be generated for future clustering.

Secondly, having established the feature words list, users will obtain clustering information for each article they input subsequently.

Thirdly, users will also be capable to append new categories even after the generation of featured words.

Based on our design, we believe that programmers should have a clear idea about what they are going to implement and how to realize all the functions.

Appendix I

Classifier
main(String[]):void
CalculateSimilarity
BuildVectorForInputFile
BuildDocVector
BuildWordCountMatrix
CalculateIG
FeatureSelection
SingleFileWordCount
BuildDictionary
ReadContent
0..*
wordCountSet