References

[1] Christopher D. Manning and et al (2008) An Interdiction to Information Retrieval [Online] Available: http://nlp.stanford.edu/IR-book/information-retrieval-book.html, Accessed [10-JAN-2011]

[2] ANTLR parser generator [Online]. Available: http://www.antlr.org Accessed [10-JAN-2011]

[3] Christian Arwin and et al. (2006), Plagiarism Detection Across Programming Languages, Research and Practice in Information Technology, Vol. 48. ACSC2006, Hobart, Tasmania, Australia

[4] Eclipse open source cross multi-language software development environment [Online] Available: http://www.eclipse.org, Accessed [10-JAN-2011]

[5] EthemAlpapaydin (2004), An Introduction to Machine Learning, MIT Press, Cambridge, MA, USA

[6] Haibiao Ding and et al. (2004), Extraction of Java Program Fingerprints for Software Authorship Identification, the Journal of Systems and Software 72, pp 49-57

[7] K. Arnold, J. Gosling, and D. Holmes, The Java(TM) Programming Language, Prentice Hall, 4 ed., Aug. 2005.

[8]Jeong-HoonJi and et al. (2007), A Source Code Linearization Techniques for Detecting Plagiarized Programs, ITiCSE'07 June 23-27, Dundee, Scotland, United Kingdom, pp 73-77

[9] Robert Lange and et al. (2007), Using Code Metric Histograms and Genetic Algorithms to Perform Author Identification for Software Forensics, GECCO'07 July 7-11, London, England , United Kingdom, pp 2082-2089

[10] Steve Engels and et al. (2007), Plagiarism Detection Using Feature-Based Neural Networks, SIGCSE'07, March 7-10, Covington, Kentucky, USA , pp 34-38

[11] Stuart J. Russell and et al (2003), Artificial Intelligence A Modern Approach, Second Edition, Prentice Hall, Inc.

[12] Wikipedia the Free Encyclopedia [Online] Available: http://www.wikipedia.org/, Accessed [10-JAN-2011]

Appendix A

Machine Learning Algorithms Used in the Research

Appendix A discusses the various machine learning algorithms used in the research system as well as the results obtained from those algorithms. All the algorithms are presented in pseudo code format. When constructing the system these algorithms were converted into the Java [7] programming language.

1. Multinomial Naïve Bayes Algorithm

1: $TRAINMULTINIMIALNB(\mathbb{C}, \mathbb{D})$
2: $V \leftarrow EXTRACTVOCABULARY(\mathbb{D})$
3: $N \leftarrow COUNTDOCS(\mathbb{D})$
4: **for each** $c \in \mathbb{C}$
5: **do** $N_c \leftarrow COUNTDOCSINCLASS(\mathbb{D}, c)$
6: $prior[c] \leftarrow N_c / N$
7: $text_c \leftarrow CONCATENATETEXTOFALLDOCSINCLASS(\mathbb{D}, c)$
8: **for each** $t \in V$
9: **do** $T_{ct} \leftarrow COUNTTOKENSOFTERM(text_c, t)$
10: **for each** $t \in V$
11: **do** $condprob[t][c] \leftarrow \dfrac{T_{ct} + 1}{\sum_{t'}(T_{ct'} + 1)}$
12: **return** $V, prior, condprob$

1: $APPLYMUKTINOMIALNB(\mathbb{C}, V, prior, condprob, d)$
2: $W \leftarrow EXTRACTTOKENSFROMDOC(V, d)$
3: **for each** $c \in \mathbb{C}$
4: **do** $score[c] \leftarrow \log(prior[c])$
5: **for each** $t \in W$
6: **do** $score[c] \mathrel{+}= \log(condprob[t][c])$
7: **return** $\arg\ max_{c \in \mathbb{C}}\ score[c]$

Figure A.1 - Naïve Bayes Multinomial Algorithm

Source: In interdiction to Information Retrieval, Christopher D. Manning and et al

Figure A.2 shows the confusion matrix for a training set of 200 source code files.

CONFUSION MATRIX FOR MULTINOMIAL NAIVE BAYSE CLASSIFIER

		Predicted Author											
		#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	Tot:	Suc:	%
	#1	9	0	0	0	0	0	0	0	0	1	10	9	90
	#2	0	9	0	0	0	1	0	0	0	0	10	9	90
	#3	0	0	8	0	0	0	0	0	2	0	10	8	80
	#4	0	0	0	10	0	0	0	0	0	0	10	10	100
Actual	#5	0	0	0	0	8	0	1	0	0	1	10	8	80
Author	#6	0	0	0	0	0	10	0	0	0	0	10	10	100
	#7	0	0	0	0	0	0	10	0	0	0	10	10	100
	#8	0	0	0	0	0	0	0	10	0	0	10	10	100
	#9	0	0	0	0	0	0	0	0	10	0	10	10	100
	#10	0	0	0	4	0	0	0	1	0	5	10	5	50

TOTAL FILES CLASSIFIED : 100
TOTAL SUCCESSES : 89
TOTAL FAILURES : 11
SUCCESS RATE : 89

Figure A.2 - Confusion matrix for multinomial naïve bayes for 200 source code files

Figure A.3 shows the confusion matrix for a training set of 300 source code files.

CONFUSION MATRIX FOR MULTINOMIAL NAIVE BAYSE CLASSIFIER

		Predicted Author											
		#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	Tot:	Suc:	%
	#1	10	0	0	0	0	0	0	0	0	0	10	10	100
	#2	0	5	0	1	0	0	0	0	0	4	10	5	50
	#3	0	0	9	0	0	0	0	0	1	0	10	9	90
	#4	0	0	0	7	0	0	0	0	0	3	10	7	70
Actual	#5	0	0	0	0	10	0	0	0	0	0	10	10	100
Author	#6	0	0	0	0	0	10	0	0	0	0	10	10	100
	#7	0	0	0	0	0	0	10	0	0	0	10	10	100
	#8	0	1	0	0	0	0	0	9	0	0	10	9	90
	#9	0	0	0	0	0	0	0	1	9	0	10	9	90
	#10	0	1	0	0	0	1	0	0	0	8	10	8	80

TOTAL FILES CLASSIFIED : 100
TOTAL SUCCESSES : 87
TOTAL FAILURES : 13
SUCCESS RATE : 87

Figure A.3 - Confusion matrix for multinomial naïve bayes for 300 source code files

Figure A.4 shows the confusion matrix for a training set of 400 source code files.

CONFUSION MATRIX FOR MULTINOMIAL NAIVE BAYSE CLASSIFIER

		Predicted Author											
		#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	Tot:	Suc:	%
Actual Author	#1	10	0	0	0	0	0	0	0	0	0	10	10	100
	#2	0	10	0	0	0	0	0	0	0	0	10	10	100
	#3	0	0	10	0	0	0	0	0	0	0	10	10	100
	#4	0	0	0	8	0	0	1	0	0	1	10	8	80
	#5	0	0	0	0	8	0	1	0	0	1	10	8	80
	#6	0	0	0	0	0	10	0	0	0	0	10	10	100
	#7	0	0	0	0	0	0	9	0	1	0	10	9	90
	#8	0	0	0	0	0	0	0	9	1	0	10	9	90
	#9	0	0	0	0	0	0	0	0	10	0	10	10	100
	#10	0	0	0	4	0	1	0	0	0	5	10	5	50

TOTAL FILES CLASSIFIED : 100
TOTAL SUCCESSES : 89
TOTAL FAILURES : 11
SUCCESS RATE : 89

Figure A.4 - Confusion matrix for multinomial naïve bayes for 400 source code files

Figure A.5 shows the confusion matrix for a training set of 500 source code files.

CONFUSION MATRIX FOR MULTINOMIAL NAIVE BAYSE CLASSIFIER

		Predicted Author											
		#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	Tot:	Suc:	%
Actual Author	#1	10	0	0	0	0	0	0	0	0	0	10	10	100
	#2	0	9	0	1	0	0	0	0	0	0	10	9	90
	#3	0	0	9	0	0	0	0	1	0	0	10	9	90
	#4	0	0	0	8	0	0	0	0	0	2	10	8	80
	#5	0	0	0	0	8	0	0	0	0	2	10	8	80
	#6	0	0	0	0	0	10	0	0	0	0	10	10	100
	#7	0	0	0	0	0	0	9	0	1	0	10	9	90
	#8	0	0	0	0	0	0	0	9	1	0	10	9	90
	#9	0	0	0	0	0	0	0	0	10	0	10	10	100
	#10	0	0	0	0	0	0	0	0	0	10	10	10	100

TOTAL FILES CLASSIFIED : 100
TOTAL SUCCESSES : 92
TOTAL FAILURES : 8
SUCCESS RATE : 92

Figure A.5 - Confusion matrix for multinomial naïve bayes for 500 source code files

Figure A.6 shows the confusion matrix for a training set of 600 source code files.

CONFUSION MATRIX FOR MULTINOMIAL NAIVE BAYSE CLASSIFIER

		Predicted Author											
		#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	Tot:	Suc:	%
Actual Author	#1	10	0	0	0	0	0	0	0	0	0	10	10	100
	#2	0	9	0	0	0	0	0	0	1	0	10	9	90
	#3	0	0	9	0	0	0	0	0	1	0	10	9	90
	#4	0	0	0	8	0	0	0	0	0	2	10	8	80
	#5	0	0	0	0	7	0	1	0	0	2	10	7	70
	#6	0	0	0	0	0	10	0	0	0	0	10	10	100
	#7	0	0	0	0	0	0	10	0	0	0	10	10	100
	#8	0	0	0	0	0	2	0	7	1	0	10	7	70
	#9	0	0	0	0	0	0	0	0	10	0	10	10	100
	#10	0	0	0	0	0	0	0	0	1	9	10	9	90

TOTAL FILES CLASSIFIED : 100
TOTAL SUCCESSES : 89
TOTAL FAILURES : 11
SUCCESS RATE : 89

Figure A.6 - Confusion matrix for multinomial naïve bayes for 600 source code files

Figure A.7 shows the confusion matrix for a training set of 700 source code files.

CONFUSION MATRIX FOR MULTINOMIAL NAIVE BAYSE CLASSIFIER

		Predicted Author											
		#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	Tot:	Suc:	%
Actual Author	#1	10	0	0	0	0	0	0	0	0	0	10	10	100
	#2	0	10	0	0	0	0	0	0	0	0	10	10	100
	#3	0	0	8	0	0	0	0	1	1	0	10	8	80
	#4	0	0	0	10	0	0	0	0	0	0	10	10	100
	#5	0	0	0	0	10	0	0	0	0	0	10	10	100
	#6	0	0	0	0	0	9	0	0	0	1	10	9	90
	#7	0	0	0	0	0	0	10	0	0	0	10	10	100
	#8	0	0	0	0	0	1	0	9	0	0	10	9	90
	#9	1	0	0	0	0	0	0	0	9	0	10	9	90
	#10	0	0	0	0	1	0	1	0	1	7	10	7	70

TOTAL FILES CLASSIFIED : 100
TOTAL SUCCESSES : 92
TOTAL FAILURES : 8
SUCCESS RATE : 92

Figure A.7 - Confusion matrix for multinomial naïve bayes for 700 source code files

2. Bernoulli Naïve Bayes Algorithm

1: $TRAINBERNOULLINB(\mathbb{C}, \mathbb{D})$
2: $V \leftarrow$ ***EXTRACTVOCABULARY***$(\mathbb{D})$
3: $N \leftarrow$ ***COUNTDOCS***$(\mathbb{D})$
4: ***for each*** $c \in \mathbb{C}$
5: ***do*** $N_c \leftarrow COUNTDOCSINCLASS(\mathbb{D}, c)$
6: $prior[c] \leftarrow {}^{N_c}/_{N}$
7: ***for each*** $t \in V$
8: ***do*** $N_{ct} \leftarrow COUNTDOCSINCLASSCONTAININGTERM(\mathbb{D}, c, t)$
9: $condprob[t][c] \leftarrow {}^{(N_{ct}+1)}/_{(N_c+2)}$
10: ***return*** $V, prior, condprob$

1: $APPLYBERNOULLINB(\mathbb{C}, V, prior, condprob, d)$
2: $V_c \leftarrow EXTRACTTERMSFROMDOC(V, d)$
3: ***for each*** $c \in \mathbb{C}$
4: ***do*** $score[c] \leftarrow \log prior[c]$
5: ***for each*** $t \in V$
6: ***do if*** $t \in V_d$
7: ***then*** $socre[c] += \log condprob[t][c]$
8: ***else*** $socre[c] += \log(1 - condprob[t][c])$
9: ***return*** $\arg max_{c \in \mathbb{C}} score[c]$

Figure A.8 - Naïve Bayes Bernoulli Algorithm.

Source: In interdiction to Information Retrieval, Christopher D. Manning and et al

Figure A.9 shows the confusion matrix for a training set of 200 source code files

CONFUSION MATRIX FOR BERNOULLI NAIVE BAYSE CLASSIFIER

		Predicted Author											
		#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	Tot:	Suc:	%
Actual Author	#1	10	0	0	0	0	0	0	0	0	0	10	10	100
	#2	0	7	0	0	0	0	3	0	0	0	10	7	70
	#3	0	0	6	0	0	0	1	1	2	0	10	6	60
	#4	0	0	0	8	0	0	2	0	0	0	10	8	80
	#5	0	0	0	0	5	0	5	0	0	0	10	5	50
	#6	0	0	0	0	0	9	1	0	0	0	10	9	90
	#7	0	0	0	1	0	0	7	0	0	2	10	7	70
	#8	0	0	0	0	0	0	0	9	1	0	10	9	90
	#9	0	0	0	0	0	0	3	2	5	0	10	5	50
	#10	0	0	0	0	0	1	2	0	2	5	10	5	50

TOTAL FILES CLASSIFIED : 100
TOTAL SUCCESSES : 71
TOTAL FAILURES : 29
SUCCESS RATE : 71

Figure A.9 Confusion matrix for Bernoulli naive bayes for 200 source code files

Figure A.10 shows the confusion matrix for a training set of 300 source code files

CONFUSION MATRIX FOR BERNOULLI NAIVE BAYSE CLASSIFIER

		Predicted Author											
		#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	Tot:	Suc:	%
Actual Author	#1	8	0	0	0	0	0	1	0	0	1	10	8	80
	#2	0	9	0	0	0	0	0	0	0	1	10	9	90
	#3	0	0	8	0	0	0	0	1	1	0	10	8	80
	#4	0	0	0	9	0	0	0	0	0	1	10	9	90
	#5	0	0	0	0	8	0	1	0	0	1	10	8	80
	#6	0	0	0	0	0	9	1	0	0	0	10	9	90
	#7	0	0	2	0	0	0	7	1	0	0	10	7	70
	#8	0	0	0	0	0	0	1	9	0	0	10	9	90
	#9	0	0	0	0	0	0	1	4	5	0	10	5	50
	#10	0	0	0	1	0	0	0	1	1	7	10	7	70

TOTAL FILES CLASSIFIED : 100
TOTAL SUCCESSES : 79
TOTAL FAILURES : 21
SUCCESS RATE : 79

Figure A.10 - Confusion matrix for Bernoulli naïve bayes for 300 source code files

Figure A.11 shows the confusion matrix for a training set of 400 source code files

CONFUSION MATRIX FOR BERNOULLI NAIVE BAYSE CLASSIFIER

		Predicted Author											
		#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	Tot:	Suc:	%
Actual Author	#1	9	0	0	0	0	0	1	0	0	0	10	9	90
	#2	0	10	0	0	0	0	0	0	0	0	10	10	100
	#3	0	0	5	0	0	0	1	2	2	0	10	5	50
	#4	0	0	0	10	0	0	0	0	0	0	10	10	100
	#5	0	0	0	0	9	0	1	0	0	0	10	9	90
	#6	0	0	0	0	0	9	0	0	1	0	10	9	90
	#7	0	0	1	0	0	0	9	0	0	0	10	9	90
	#8	0	0	0	0	0	1	1	3	5	0	10	3	30
	#9	0	0	0	0	0	0	0	3	6	1	10	6	60
	#10	0	0	0	0	0	0	2	1	0	7	10	7	70

TOTAL FILES CLASSIFIED : 100
TOTAL SUCCESSES : 77
TOTAL FAILURES : 23
SUCCESS RATE : 77

Figure A.11 - Confusion matrix for Bernoulli naïve bayes for 400 source code files

Figure A.12 shows the confusion matrix for a training set of 500 source code files

CONFUSION MATRIX FOR BERNOULLI NAIVE BAYSE CLASSIFIER

		Predicted Author											
		#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	Tot:	Suc:	%
Actual Author	#1	9	0	0	0	0	0	0	1	0	0	10	9	90
	#2	0	9	0	0	0	0	1	0	0	0	10	9	90
	#3	0	0	8	0	0	0	1	1	0	0	10	8	80
	#4	0	0	0	8	0	0	1	0	0	1	10	8	80
	#5	0	0	0	0	9	0	1	0	0	0	10	9	90
	#6	0	0	0	0	0	9	0	0	1	0	10	9	90
	#7	0	0	0	0	0	0	6	1	0	3	10	6	60
	#8	0	0	0	0	0	0	0	8	2	0	10	8	80
	#9	0	0	1	0	0	0	0	4	5	0	10	5	50
	#10	0	0	0	0	0	0	3	0	1	6	10	6	60

TOTAL FILES CLASSIFIED : 100
TOTAL SUCCESSES : 77
TOTAL FAILURES : 23
SUCCESS RATE : 77

Figure A.12 - Confusion matrix for Bernoulli naïve bayes for 500 source code files

Figure A.13 shows the confusion matrix for a training set of 600 source code files

CONFUSION MATRIX FOR BERNOULLI NAIVE BAYSE CLASSIFIER

		Predicted Author											
		#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	Tot:	Suc:	%
Actual Author	#1	9	0	0	0	0	0	0	0	0	1	10	9	90
	#2	0	10	0	0	0	0	0	0	0	0	10	10	100
	#3	0	0	8	0	0	0	1	1	0	0	10	8	80
	#4	0	0	1	8	0	0	0	0	0	1	10	8	80
	#5	0	0	0	0	8	0	2	0	0	0	10	8	80
	#6	0	2	0	0	0	7	0	0	1	0	10	7	70
	#7	0	0	0	0	0	0	9	0	0	1	10	9	90
	#8	0	0	0	0	0	0	0	8	2	0	10	8	80
	#9	0	0	0	0	0	0	3	4	3	0	10	3	30
	#10	0	0	0	0	0	0	0	1	3	6	10	6	60

TOTAL FILES CLASSIFIED : 100
TOTAL SUCCESSES : 76
TOTAL FAILURES : 24
SUCCESS RATE : 76

Figure A.13 - Confusion matrix for Bernoulli naïve bayes for 600 source code files

Figure A.14 shows the confusion matrix for a training set of 700 source code files

CONFUSION MATRIX FOR BERNOULLI NAIVE BAYSE CLASSIFIER

		Predicted Author											
		#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	Tot:	Suc:	%
Actual Author	#1	7	0	1	0	0	0	1	0	0	1	10	7	70
	#2	0	10	0	0	0	0	0	0	0	0	10	10	100
	#3	0	0	5	0	0	0	1	4	0	0	10	5	50
	#4	0	0	0	10	0	0	0	0	0	0	10	10	100
	#5	0	0	0	0	7	0	3	0	0	0	10	7	70
	#6	0	1	0	0	0	9	0	0	0	0	10	9	90
	#7	0	0	0	0	0	0	9	0	0	1	10	9	90
	#8	0	0	0	0	0	0	0	10	0	0	10	10	100
	#9	0	0	0	0	0	0	3	5	2	0	10	2	20
	#10	0	0	0	0	0	0	0	3	0	7	10	7	70

TOTAL FILES CLASSIFIED : 100
TOTAL SUCCESSES : 76
TOTAL FAILURES : 24
SUCCESS RATE : 76

Figure A.14 - Confusion matrix for Bernoulli naïve bayes for 700 source code files

3. k-Nearest Neighbor (kNN) Algorithm

$1: TRAIN - kNN(\mathbb{C}, \mathbb{D})$

$2: \quad \mathbb{D}' \leftarrow PREPROCESS(\mathbb{D})$

$3: \quad k \leftarrow SELECT - K(\mathbb{C}, \mathbb{D}')$

$4: \quad \boldsymbol{return} \mathbb{D}', k$

$1: APPLY - kNN(\mathbb{C}, \mathbb{D}', k, d)$

$2: \quad S_k \leftarrow COMPUTENEARESTNEIGHBORS(\mathbb{D}', k, d)$

$3: \quad \boldsymbol{for\ each} c_j \in \mathbb{C}$

$4: \quad \boldsymbol{do} p_j \leftarrow {|S_k \cap c_j|}/{k}$

$5: \quad \boldsymbol{return}$ arg $max_j p_j$

Figure A.15 - kNN Algorithm

Source: In interdiction to Information Retrieval, Christopher D. Manning and et al

4. Ada Boost Algorithm

$1: \boldsymbol{function}\ ADABOOST(examples, L, M)$

$2: \boldsymbol{inputs}: examples, a\ set\ of\ N\ labelled\ examples$

$L, a\ learning\ algorithm$

$M, the\ number\ of\ hypertheses\ in\ the\ ensemble$

$3: \boldsymbol{local\ variables}: \boldsymbol{w}, a\ vector\ of\ N\ example\ weights, initially\ {1}/{N}$

$\boldsymbol{h}, a\ vector\ of\ M\ hypertheses$

$\boldsymbol{z}, a\ vector\ of\ M\ hypertheses\ weights$

$4: \boldsymbol{for}\ m = 1\ \boldsymbol{to}\ M\ \boldsymbol{do}$

$5: \quad \boldsymbol{h}[m] \leftarrow L(examples, w)$

$6: \quad error \leftarrow 0$

$7: \quad \boldsymbol{for}\ j = 1\ \boldsymbol{to}\ N\ \boldsymbol{do}$

8: $\boldsymbol{if}\ \boldsymbol{h}[m](x_j) \neq y_i\ \boldsymbol{then}\ error \leftarrow error + \boldsymbol{w}[j]$

9: $\boldsymbol{for}\ j = 1\ \boldsymbol{to}\ N\ \boldsymbol{do}$

10: $\boldsymbol{if}\ \boldsymbol{h}[m](x_j) = y_j\ \boldsymbol{then}\ \boldsymbol{w}[j] \leftarrow \boldsymbol{w}[j] * {}^{error}/_{(1-error)}$

11: $\boldsymbol{w} \leftarrow NORMALIZE(\boldsymbol{w})$

12: $\boldsymbol{z}[n] \leftarrow {}^{log(1-error)}/_{error}$

13: $\boldsymbol{return}\ WEIGHTED - MAJORITY(\boldsymbol{h}, \boldsymbol{z})$

Figure A.16 - Ada Boost Algorithm.

Source: Stuart J. Russell and et al (2003), Artificial Intelligence a Modern Approach

Appendix B

Contents of the CD-ROM

1. Full source code of the implementation: CD-ROM contains the complete Java source code of the project including the Eclipse [4] project file.
2. The training dataset used for the research.
3. The soft-copy of the dissertation.

File or Directory	Description
/readme.txt	This file describes the contents of the CD.
/dataset/*.*	The training and validation dataset used for the research.
/src/*.*	The complete Java [7] source code of the project including Eclipse [4] project file
/Thesis/*.*	The thesis and related references.