REF:JKU/2/83/022

Jomo Kenyatta University of Agriculture and Technology (JKUAT)

School of Computing and Information Technology (SCIT)

Department of Computing

`Final Year Project Report`

Student Name:_____	Reg. No:_____
Course:_____	
Title/Research Topic:_____	
Supervisor 1:_____	Sign:_____
Supervisor 2:_____	Sign:_____
Supervisor 3:_____	Sign:_____
Date Submitted:_____	

Abstract

Cheating on university assignments and essays has become easier as a result of thousands of papers which can be found online. Anti-plagiarism software stands a chance of catching the cheaters, but they come short in scenarios where the cheater has hired a custom essay writer.

Stylometry assumes that there is an unconscious aspect to an author's writing style that possesses quantifiable and distinctive features and cannot be consciously manipulated. These characteristic features of an author should be frequent, easily quantifiable and relatively immune to conscious control. It is these features that distinguish authors writing in similar topics.

This project will focus on attempts to determine the authorship of essays based on the writer's styles of writing. This will serve to assist lecturers to ascertain that the essays handed in by the students are their own work, and not someone else.

Acknowledgment

First of all, I would like to dedicate my special thanks to God the Almighty; By His grace I have been able to accomplish this project.

Special thanks also go to my supervisors Professor Waweru Mwangi and Mr. John Wainaina, who guided me with a valuable information, references and ideas to finish this project. Working with them has been a memorable experience for me.

My appreciation also goes to other Computer Science department lecturers and staff, thank you for all the support and help during my study.

Final thanks to my mom, dad, kin and friends for moral and financial support and my classmates.

Contents

List of Figures

List of Tables

Outline of final year project report writing.

1. Introduction
2. Literature Review
3. Research Methodology
4. System Design
5. System Implementation
6. System Testing
7. Results and Analysis
8. Conclusion and Recommendations/Future Work
9. Bibliography
10. Appendices

1 INTRODUCTION

1.1 Background

1.1.1 Stylometry

Stylometry refers to the study of the unique linguistic styles and writing behaviours of individuals so as to determine authorship. It subsumes statistical methods for quantifying an author's unique writing style. By constructing and comparing stylometric models for different text segments, passages that are stylistically different from others, hence potentially plagiarized, can be detected.

Stylometry uses pattern recognition, statistical analysis, and artificial intelligence techniques. It analyses the text in the documents or essays by using various parameters. These include the topic of the document and its content, for example word frequencies. It also identifies pattern in common parts of speech.

Stylometry has its origin traced back to the mid-19th century where Augustus de Morgan, an English logician, suggested that word length could be an indicator of ownership. Later on in 1964, Mosteller and Wallace, two American statisticians, made an even bigger impact in this area when they decided to use word frequencies to investigate the mystery of the authorship of The Federal Papers.

1.1.2 Classification

The process of recognizing patterns and classifying data accordingly has been gaining interest from a long time and human beings have developed highly sophisticated skills for sensing from their environment and take actions according to what they observe. So a human can recognize the faces without worrying about the varying illuminations, facial rotation, facial expressions, and facial biometrical changes and also occluded face images. But if the point of implementing such recognition artificially came, then it becomes a very complex task. The fields of artificial intelligence have made this complex task possible by making machines as intelligent as human to recognize patterns in varying environmental conditions. Such a branch of artificial intelligence is known as pattern recognition.

Pattern Recognition provides the solution to a lot of problems that fall under the category of either recognition or classification, such as speech recognition, face recognition, classification of handwritten characters and medical diagnosis.

1.1.3 Classification Algorithms

The following are the classification algorithms according to Sharma [3];

1. *Linear discriminant analysis (LDA) :* It is used to find a linear combination of features which characterizes or separates two or more classes of objects or events. LDA is a parametric approach in supervised learning technique. It was initially used for dimensionality reduction and feature extraction, and later moved for classification purpose.

2. *Quadratic Discriminant Analysis (QDA) :* It is used in machine learning and statistical classification to separate measurements of two or more classes of objects or events by a quadric surface. It is a more general version of a linear classifier. QDA is a parametric approach in supervised learning which models the likelihood of each class as a Gaussian distribution, then uses the posterior distributions to estimate the class for a given test point.

3. *Maximum entropy classifier (multinomial logistic regression) :* In statistics, a maximum entropy classifier model is a regression model which generalizes logistic regression by allowing more than two discrete outcomes. This forms a model that is used to predict the probabilities of the different possible outcomes of a categorically distributed dependent variable, given a set of independent variables (which may be real-valued, categorical-valued etc.).The actual goal of the multinomial logistic regression model is to predict the categorical data.

4. *Decision trees :* It is considered to be a decision support tool that uses a tree-like structure or model of decisions and all its possible consequences. It is one way to display an algorithm. These trees are basically used in operations research, mostly in decision analysis, to help identify a strategy most likely to reach a goal.In this process, a decision tree and the closely related influence diagram is used as a visual and analytical decision support tool where the expected values of competing alternatives are calculated.

5. *Kernel Estimation and K-nearest neighbor :* In the field of pattern recognition, the k-nearest neighbor algorithm (k-NN) is a method for classifying objects based on closest training examples in the feature space. K-NN is a type of example-based learning, or lazy learning where the function is only approximated locally and all the computation is deferred until classification. This algorithm is one of the simplest machine learning algorithm in which an object is classified using a majority vote of its neighbors and the object is then assigned to the class which is most common amongst its k-nearest neighbors.

6. *Naive Bayes classifier :* Naive Bayes classifier is a simple, probabilistic and statistical classifier which is based on Bayes theorem (from Bayesian statistics) with strong (naive) independence assumptions and maximum posteriori hypothesis. As Bayesian classifiers are statistical in nature, they can predict the probability of a given sample belonging to a particular class. The underlying probability model to this classifier can be termed more appropriately as an "independent feature model" because a naive Bayes classifier assumes that the effect of an attribute value on a given class is independent of the values of the other attributes. Such an assumption is called class conditional independence. It is made to simplify the computation involved and, in this sense, is considered "naive".

7. *Artificial Neural Networks :* It is an interconnected network of a group of artificial neurons.An artificial neuron can be considered as a computational model which is inspired by the natural neurons present in human brain. Unlike natural neurons, the complexity is highly abstracted when modeling artificial neurons. These neurons basically consist of inputs (like synapses), which are further multiplied by a parameter known as weights (strength of each signals), and then computed by a mathematical function which determines the activation of the neuron. After this there is another function that computes the output of the artificial neuron (sometimes in dependence of a certain threshold). Thus the artificial networks are formed by combining these artificial neurons to process information.

8. *Support Vector Machine :* A Support Vector Machine (SVM) performs classification by constructing an N-dimensional hyperplane that optimally separates the data into two categories. A support vector machine (SVM) is used in computer science for a set of related supervised learning methods that analyze input data and learns from it and then use it for performing classification and regression analysis. The standard SVM is a two-class SVM which takes a set of input data and predicts the possible class, for each input, among the two possible classes the input is a member of, which makes it a non-probabilistic binary linear classifier. Given the set of training examples where each one of them is marked as belonging to one of the two categories, the SVM training algorithm builds a model that assigns new examples into one category or the other.

1.1.4 Pattern Recognition

Pattern recognition refers to a branch of Artificial Intelligence that is concerned with the study of methods and algorithms for putting data objects into categories by recognizing patterns in them. It encloses sub disciplines like discriminant analysis, feature extraction, error estimation, cluster analysis, grammatical inference and parsing.

A pattern is a set of objects or phenomena or concepts where the elements of the set are similar to one another in certain aspects [3]. Pattern category on the other hand is the collection of similar, not necessarily identical object. Often, individual patterns may be grouped into a category based on their common properties; the resultant is also a pattern and is often called a pattern category.

Below is a design model entailing the steps in a pattern recognition system [3]

1. *Data acquisition and pre-processing:* Here, the data from the surrounding environment is taken as the input and given to the pattern recognition system. The data is pre-processed by extracting pattern of interest from the backgrounds so as to make the input data readable by the system.
2. *Feature extraction:* Here, the relevant features from the processed data are extracted. These relevant features collectively form entity of objects to be recognized or classified.
3. *Decision making:* Here the desired operation of classification or recognition is done upon the descriptor of extracted features

Pattern recognition techniques that are used in Stylometry include supervised learning, unsupervised learning, linear discriminant analysis, quadratic discrimi-

nant analysis, maximum entropy classifier, decision trees, kernel estimation and K-nearest neighbour, neural networks and support vector machine.

1.2 Problem Statement

Plagiarism detectors do not actually detect plagiarism. Instead, what they actually detect is sections of identical text. A plagiarism detection service looks for matching strings of words between the document its looking at and the ones it has in its index. This makes anti-plagiarism softwares limited in the following ways;

1. *Non-Verbatim Plagiarism :* Plagiarism that involves the rewriting, translating or otherwise redrafting the text can't be detected. This can be difficult to get away with as most plagiarism detectors are extremely sensitive, but since plagiarism detectors don't analyze the content of the work, just the words, it can't see if you lifted the idea or information if you didn't also lift the words. This is a common problem in academia, which treats this kind of plagiarism equally as seriously as verbatim plagiarism.
2. *Common Phrasing/Attributed Use :* Though many plagiarism checkers will make an attempt to separate out attributed use, given the variety of attribution styles it isn't always possible. Also, given how common some phrases are in the English language, many plagiarism checkers will report matches that are actually just coincidence.

James Heather, senior lecturer in computing at the University of Surrey, has revealed that plagiarism detection systems such as Turnitin that are routinely used by universities are open to simple cheats allowing students to evade detection when submitting copied material. Students aware of this loophole could get around the system by converting a plagiarised essay to PDF format, he says, and then altering the corresponding "character map" - a map of the sequence of characters used in the text. Although the text would remain visually unaltered, extracts tested by the plagiarism software would be garbled, and so matches would not be detected. Or, he says, students could rearrange character codes, or "glyphs", in the PDF so they no longer correspond to the alphabet and "the link between the text and its printed representation will be broken". In this scenario a tutor could print out and read the essay, but the computer running the detection software would scan nonsense [8].

There is also the social implication of the problem that needs to be considered. The grades awarded to some students in universities in their assignments, for instance, do not reflect the actual knowledge that they have. The reason is that

some students submit assignments that they copied and pasted from an online resource, or they did not do it themselves, for assessment and grading by their lecturers. There are software applications that can detect the former act easily, and some of them are freely available. This problem statement will focus on the latter act.

There are people, who may or may not include other university students, who charge a sum of money to students who want their assignments and essays done for them. This is not unheard of. There are even websites that act as middle-men between the students who need their assignments done, and people who are willing to do them for a fee.

The problem with this fast growing culture in the universities is that it gives the lectures a false impression of the students' understanding of the unit/lecture/course. This leads to the awarding of grades that do not reflect the actual situation. These false grades affect sincere hardworking students negatively.

An even bigger impact of this problem will be the shame cast on the university if the students participating in this trend, upon graduation and being hired at organizations that require some of the knowledge gained while studying at the university level, embarrass themselves out there. If a person finds an easy way out of a problem that works for him/her, then it will be difficult for him/her to quit. The same can be said for students in this scenario.

1.3 Proposed Solution

The solution is a pattern recognition system that can identify unique features in a student's writing style and analyse it so as to determine whether the essays and documents handed in to the lecturer for assessment are genuinely the works of the student.

1.4 Justification

Moore [1] writes that according to a survey by CBC News of 42 Canadian universities, more than 7,000 students were formally disciplined for academic cheating in 2011-2012. Plagiarism accounted for roughly 50% of the cases. Some of the universities said that they use some form of anti-plagiarism software. However, this software is no good when it comes to detecting a custom essay written by someone else (for profit).

1.5 Research Questions

1. How does Stylometry work, and how can it be achieved?
2. What has been done in the field of Stylometry and pattern recognition in the past, and what were the findings?
3. How to determine the measure of accuracy of the stylometric system?
4. How much data is required for a stylometric profile to be created?

1.6 Objectives

1.6.1 Main objective

To create a pattern recognition system to determine the authorship of essays submitted to it.

1.6.2 Specific objectives

1. To analyse literature in the area of pattern recognition. This includes comparing the measures of accuracy between decision trees and neural networks.
2. To analyse literature in the area of Stylometry, and how it can be achieved.
3. To model and evaluate a pattern recognition system that will determine authorship.
4. To arrive at a reasonable allowable deviation from the mean that the results of the stylometric analysis of the essays can deviate.
5. To investigate the amount of data required for a stylometric profile of the student to be created.

1.7 Scope

This research and project will only be limited to decision trees for pattern recognition.

Table 1: Source Table

Description	**Amount (Ksh.)**
Printing, Copying and Binding	500.00
Internet Data Bundle	3,000.00
Modem	2,000.00
Transportation Costs	1,000.00
Communication Costs	1,000.00
Flash Disk	1,000.00
Open Source Tools	Free
Total	8,500.00

Table 2: Source Table

Task ID	**Task Name**	**Expected Start Date**	**Expected End Date**	**Deliverables**
1	Project Title	June 1, 2014	June 11, 2014	Draft Proposal
2	Project Proposal Presentation	June 11, 2014	June 30, 2014	Final Proposal Document
3	Literature Review Presentation	August 4, 2014	September 8, 2014	Literature Review Report
4	Project Implementation	September 8, 2014	October 20, 2014	Working Prototype
5	Project Testing and Debugging	October 20, 2014	November 3, 2014	Final Prototype
6	Documentation	June 30, 2014	November 24, 2014	Complete System with Documentation

1.8 Budget

1.9 Project Schedule

2 LITERATURE REVIEW

2.1 Introduction

Large scale real-time data became available due to advanced technology in computer systems. Data mining is the process of extracting valid and comprehensible information from such a large database so as to make significant contributions to crucial management decision processes. A particular concern in a paper by So Young Sohn [6] is the several classification algorithms used for pattern recognition in the process of data mining.

Pattern recognition has been used in the fields of stylistics, computational linguistics, and non-traditional authorship attribution to develop a possible framework for the identification of e-mail text authorship. The fields like text classification, machine learning, software forensics, and forensic linguistics also impact on the current study.

So Young Sohn [6] reviews classification algorithms in three areas; traditional statistical approach, artificial neural networks, and machine learning. Traditionally, parametric statistical approaches such as discriminant analysis have been extensively used to classify one group from others based on the associated individual characteristics. The main assumption required for the discriminant analysis is that these characteristics follow a multi-variate normal distribution with distinct means for each group and a common variance-covariance matrix. When this variance-covariance matrix assumption is met for k groups, Fisher's Linear Discriminant function is used for classification. When this assumption is violated, a Quadratic Discriminant function is estimated based on an individually estimated variance-covariance matrix.

Nonparametric classification methods do not make any distributional assumptions and classify a test case based on the training samples in the neighbourhood of the item in terms of associated individual characteristics. K-Nearest Neighbor method is a popular nonparametric method. It classifies a test case into the class that supplies the largest number of neighbours among the K-Nearest Neighbors of the case.

Artificial Neural Networks (ANNs) have been suggested as an alternative methodology for classification to which traditional statistical techniques have long been applied. Some examples of supervised learning networks are MultiLayer Perception (MLP) and Radial Basis Function (RBF). MLPs are flexible nonlinear models that can approximate virtually any function to any desired degree of accuracy. In MLP, the net input to the hidden layer is a linear combination of inputs as specified

by the weights.

Unsupervised learning explores the structure of data without guidance in the form of classification. The clusters found offer a model of the data in terms of cluster centers, sizes, and shapes. Iterative clustering algorithms such as Kohonen network, are often used.

2.2 Decision Tress

Decision trees are usually constructed beginning with the root of the tree and proceeding down to its leaves. This approach can be used for predictive or descriptive purposes. Decision tree induction is free from parametric structural assumptions that most statistical induction methods, such as discriminant analysis, are based on. Baytree is a Bayesian approach to decision trees requiring the specification of prior class probabilities and a probability model for the decision tree.

The core algorithm for building decision trees is called ID3 by J. R. Quinlan which employs a top-down, greedy search through the space of possible branches with no backtracking. ID3 uses Entropy and Information Gain to construct a decision tree.

2.2.1 The Induction Task

Quinlan [9] gives a more precise statement of the induction task in his paper. The basis of a the induction task is a universe of objects that are described in terms of a collection of attributes. Each attribute measures some important feature of an object and will be limited here to taking a (usually small) set of discrete, mutually exclusive values. For example, if the objects were Saturday mornings and the classification task involved the weather, attributes might be

outlook, with values sunny, overcast, rain

temperature, with values cool, mild, hot

humidity, with values high, normal

windy, with values true, false

Hence, a particular Saturday morning might be described as

outlook: overcast

temperature: cool

humidity: normal

windy: false

Each object in the universe belongs to one of a set of mutually exclusive classes, To simplify the following treatment, we will assume that there are only two such classes denoted P and N, although the extension to any number of classes is not difficult.

In two-class induction tasks, objects of class P and N are sometimes referred to as positive instances and negative instances, respectively, of the concept being learned.

The other major ingredient is a training set of objects whose class is known. The induction task is to develop a classification rule that can determine the class of any object from its values of the attributes. The immediate question asked here is whether or not the attributes provide sufficient information to do this. In particular, if the training set contains two objects that have identical values for each attribute and yet belong to different classes, it is clearly impossible to differentiate between these objects with reference only to the given attributes. In such a case attributes will be termed inadequate for the training set and hence for the induction task.

Table 1 below shows a small training set that uses the 'Saturday morning' attributes. Each object's value of each attribute is shown, together with the class of the object (here, class P mornings are suitable for some unspecified activity). A decision tree that correctly classifies each object in the training set is given in Figure 1. Leaves of a decision tree are class names, other nodes represent attribute-based tests with a branch for each possible outcome. In order to classify an object, we start at the root of the tree, evaluate the test, and take the branch appropriate to the outcome. The process continues until a leaf is encountered, at which time the object is asserted to belong tothe class named by the leaf. Taking the decision tree of Figure 1, this process concludes that the object which appeared as an example at the start of this section, and which is not a member of the training set, should belong to class P. Notice that only a subset of the attributes may be encountered on a particular path from the root of the decision tree to a leaf; in this case, only the outlook attribute is tested before determining the class.

If the attributes are adequate, it is always possible to construct a decision tree that correctly classifies each object in the training set, and usually there are many such correct decision trees. The essence of induction is to move beyond the training set. This means constructing a decision tree that correctly classifies not only objects from the training set but other (unseen) objects as well. In order to do this, the

decision tree must capture some meaningful relationship between an object's class and its values of the attributes. Given a choice between two decision trees, each of which is correct over the training set, it seems sensible to prefer the simpler one on the grounds that it is more likely to capture structure inherent in the problem. The simpler tree would therefore be expected to classify correctly more objects outside the training set. The decision tree of Figure 3, for instance, is also correct for the training set of Table 1, but its greater complexity makes it suspect as an 'explanation' of the training set.2

Table 1. A small training set

No.	Attributes				Class
	Outlook	Temperature	Humidity	Windy	
1	sunny	hot	high	false	N
2	sunny	hot	high	true	N
3	overcast	hot	high	false	P
4	rain	mild	high	false	P
5	rain	cool	normal	false	P
6	rain	cool	normal	true	N
7	overcast	cool	normal	true	P
8	sunny	mild	high	false	N
9	sunny	cool	normal	false	P
10	rain	mild	normal	false	P
11	sunny	mild	normal	true	P
12	overcast	mild	high	true	P
13	overcast	hot	normal	false	P
14	rain	mild	high	true	N

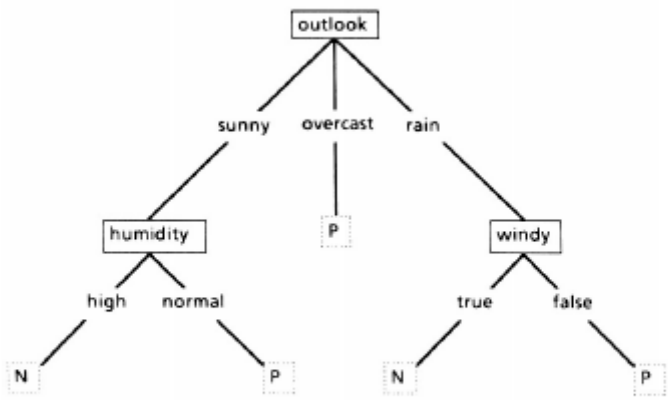


Figure 1. A Simple Decision Tree

2.2.2 ID3

One approach to the induction task above would be to generate all possible decision trees that correctly classify the training set and to select the simplest of them. The number of such trees is finite but very large, so this approach would only be feasible for small induction tasks. ID3 was designed for the other end of the spectrum, where there are many attributes and the training set contains many objects, but where a reasonably good decision tree is required without much computation. It has generally been found to construct simple decision trees, but the approach it uses cannot guarantee that better trees have not been overlooked.

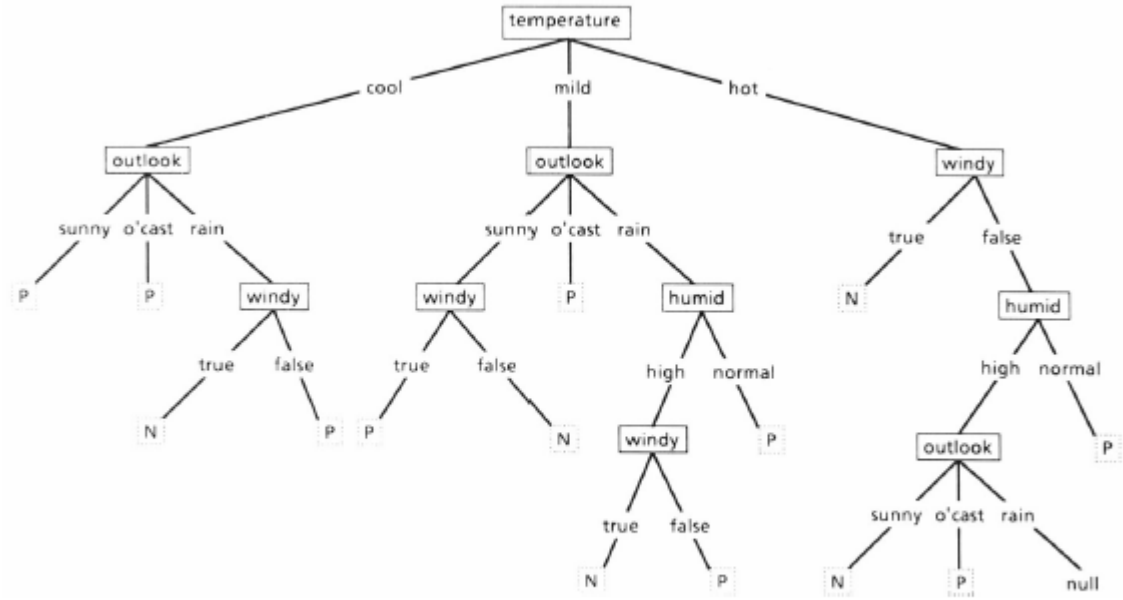


Figure 2. A Complex Decision Tree

The diagram above illustrates a complex decision tree.

The core algorithm for building decision trees is called ID3 by J. R. Quinlan which employs a top-down, greedy search through the space of possible branches with no backtracking [7].

Quinlan [9] in his paper gives a description of the ID3 algorithm. The basic structure of ID3 is iterative. A subset of the training set called the window is chosen at random and a decision tree formed from it; this tree correctly classifies all objects in the window. All other objects in the training set are then classified using the tree. If the tree gives the correct answer for all these objects then it is correct for the entire training set and the process terminates. If not, a selection of the incorrectly classified objects is added to the window and the process continues. In this way, correct decision trees have been found after only a few iterations for training sets of up to thirty thousand objects described in terms of up to 50 attributes. Empirical evidence suggests that a correct decision tree is usually found more quickly by this iterative method than by forming a tree directly from the entire training set.

The crux of the problem is how to form a decision tree for an arbitrary collection C of objects. If C is empty or contains only objects of one class, the simplest decision tree is just a leaf labelled with the class. Otherwise, let T be any test on

an object with w possible outcomes O1, O2, ... Ow. Each object in C will give one of these outcomes for T, so T produces a partition C1, C2, ... Cw of C with Ci containing those objects having outcome Oi. This is represented graphically by the tree form of Figure 2. If each subset Ci in this figure could be replaced by a decision tree for Ci, the result would be a decision tree for all of C. Moreover, so long as two or more Ci's are nonempty, each Ci is smaller than C. In the worst case, this divide-and-conquer strategy will yield single-object subsets that satisfy the one-class requirement for a leaf. Thus, provided that a test can always be found that gives a non-trivial partition of any set of objects, this procedure will always produce a decision tree that correctly classifies each object in C.

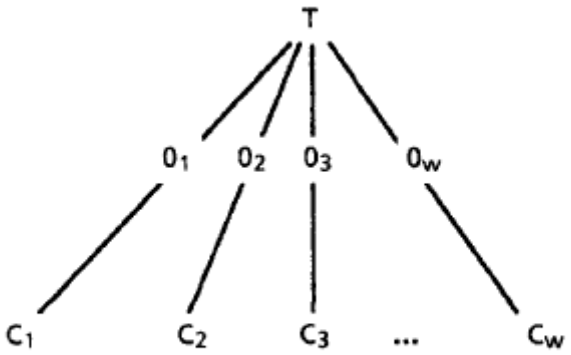


Figure 3. A Tree Structuring of the Objects in C

The choice of test is crucial if the decision tree is to be simple. For the moment, a test will be restricted to branching on the values of an attribute, so choosing a test comes down to selecting an attribute for the root of the tree. ID3 has adopted an information based method that depends on two assumptions. Let C contain p objects of class P and n objects of class N. The assumptions are:

1. Any correct decision tree for C will classify objects in the same proportion as their representation in C. An arbitrary object will be determined to belong to class P with probability p/(p + n) and to class N with probability n/(p + n).
2. When a decision tree is used to classify an object, it returns a class. A decision tree can thus be regarded as a source of a message 'P' or 'N', with the expected information needed to generate this message given by

$$I(p, n) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

If attribute A with values A1, A2, ... Av) is used for the root of the decision tree, it will partition C into (C1, C2, ... Cv where Ci contains those objects in C that have value Ai of A. Let Ci contain pi objects of class P and ni of class N. The expected information required for the subtree for Ci is I(pi, ni). The expected information required for the tree with A as root is then obtained as the weighted average

$$E(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p + n} I(p_i, n_i)$$

where the weight for the ith branch is the proportion of the objects in C that belong to Ci. The information gained by branching on A is therefore

$$\text{gain}(A) = I(p, n) - E(A)$$

A good rule of thumb would seem to be to choose that attribute to branch on which gains the most information. ID3 examines all candidate attributes and chooses A to maximize gain(A), forms the tree as above, and then uses the same process recursively to form decision trees for the residual subsets C1, C2, ... Cv.

To illustrate the idea, let C be the set of objects in Table 1. Of the 14 objects, 9 are of class P and 5 are of class N, so the information required for classification is

$$I(p, n) = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.940 \text{ bits}$$

Now consider the outlook attribute with values sunny, overcast, rain. Five of the 14 objects in C have the first value (sunny), two of them from class P and three from class N, so

$$p_1 = 2 \quad n_1 = 3 \quad I(p_1, n_1) = 0.971$$

and similarly

$$p_2 = 4 \quad n_2 = 0 \quad I(p_2, n_2) = 0$$
$$p_3 = 3 \quad n_3 = 2 \quad I(p_3, n_3) = 0.971$$

The expected information requirement after testing this attribute is therefore

$$E\ (\text{outlook}) = \frac{5}{14} I(p_1, n_1) + \frac{4}{14} I\ (p_2, n_2) + \frac{5}{14} I(p_3, n_3)$$
$$= 0.694 \text{ bits}$$

The gain of this attribute is then

$$\text{gain(outlook)} = 0.940 - \text{E(outlook)} = 0.246 \text{ bits}$$

Similar analysis gives

$$\text{gain(temperature)} = 0.029 \text{ bits}$$
$$\text{gain(humidity)} = 0.151 \text{ bits}$$
$$\text{gain(windy)} = 0.048 \text{ bits}$$

so the tree-forming method used in ID3 would choose outlook as the attribute for the root of the decision tree. The objects would then be divided into subsets according to their values of the outlook attribute and a decision tree for each subset would be induced in a similar fashion. In fact, Figure 2 shows the actual decision tree generated by ID3 from this training set.

A special case arises if C contains no objects with some particular value Aj of A, giving an empty Cj. ID3 labels such a leaf as 'null' so that it fails to classify any object arriving at that leaf. A better solution would generalize from the set C from which Cj came, and assign this leaf the more frequent class in C.

The worth of ID3's attribute-selecting heuristic can be assessed by the simplicity of the resulting decision trees, or, more to the point, by how well those trees express real relationships between class and attributes as demonstrated by the accuracy with which they classify objects other than those in the training set (their predictive accuracy). A straightforward method of assessing this predictive accuracy is to use only part of the given set of objects as a training set, and to check the resulting decision tree on the remainder.

2.3 Experiments Done — Authorship Attribution of Victorian Writers

This section gives a description of the experiment and project carried out by Ramyaa, Congzhou He and Khaled Rasheed [7].

Ramyaa and his team set out to train decision trees and neural networks to "learn" the writing style of five Victorian authors and distinguish between them based on certain features of their writing which define their style. The five Victorian authors chosen for the experiment were Jane Austen, Charles Dickens, William Thackeray, Emily Brontë and Charlotte Brontë.

The texts chosen were of the same genre and of the same period to ensure that the success of the learners would entail that texts can be classified on the style or the "textual fingerprint" of authors alone.

2.3.1 Features Used

Ramyaa and his team made use of Hanlein's empirical research [11] which has yielded a set of individual-style features, from which the 21 style indicators in the present study are derived. These include:

1. *type-token ratio:* The type-token ratio indicates the richness of an author's vocabulary. The higher the ratio, the more varied the vocabulary. It also reflects an author's tendency to repeat words.
2. *mean word length:* Longer words are traditionally associated with more pedantic and formal styles, whereas shorter words are a typical feature of informal spoken language.
3. *mean sentence length:* Longer sentences are often the indicator of carefully planned writing, while shorter sentences are more characteristic of spoken language.3
4. *standard deviation of sentence length:* The standard deviation indicates the variation of sentence length, which is an important marker of style.
5. *mean paragraph length:* The paragraph length is much influenced by the occurrence of dialogues.
6. *chapter length:* The length of the sample chapter.
7. *number of commas per thousand tokens:* Commas signal the ongoing flow of ideas within a sentence.
8. *number of semicolons per thousand tokens:* Semicolons indicate the reluctance of an author to stop a sentence where (s)he could.
9. *number of quotation marks per thousand tokens:* Frequent use of quotations is considered a typical involvement-feature.

10. *number of exclamation marks per thousand tokens:* Exclamations signal strong emotions.
11. *number of hyphens per thousand tokens:* Some authors use hyphenated words more than others.
12. *number of ands per thousand tokens:* Ands are markers of coordination, which, as opposed to subordination, is more frequent in spoken production.
13. *number of buts per thousand tokens:* The contrastive linking buts are markers of coordination too.
14. *number of howevers per thousand tokens:* The conjunct "however" is meant to form a contrastive pair with "but".
15. *number of ifs per thousands of tokens:* If-clauses are samples of subordination.
16. *number of ifs per thousands of tokens:* number of thats per thousand tokens: Most of the thats are used for subordination while a few are used as demonstratives.
17. *number of mores per thousand tokens:* 'More' is an indicator of an author's preference for comparative structure.
18. *number of musts per thousand tokens:* Modal verbs are potential candidates for expressing tentativeness. Musts are more often used non-epistemically.
19. *number of mights per thousand tokens:* Mights are more often used epistemically.
20. *number of thiss per thousand tokens:* Thiss are typically used for anaphoric reference.
21. *number of verys per thousand tokens:* Verys are stylistically significant for its emphasis on its modifiees.

2.3.2 Decision Trees — Experiments and Results

Two decision trees were made to learn the styles of the five Victorian authors. Ramyaa and his team used the See5 package by Quinlan in this experiment. The See5 packeage extends the basic ID3 algorithm of Quinlan. It infers decision trees by growing them from the root downward, greedily selecting the next best attribute for each new decision branch added to the tree. Thus, decision tree learning differs from other machine learning techniques such as neural networks, in that

attributes are considered individually rather than in connection with one another. The feature with the greatest information gain is given priority in classification. Therefore, decision trees should work very well if there are some salient features that distinguish one author from the others.

Since the attributes tested are continuous, all the decision trees are constructed using the fuzzy threshold parameter, so that the knife-edge behavior for decision trees is softened by constructing an interval close to the threshold.

The decision tree constructed without taking any other options results in an error rate of 3.3% in the training set and an error rate of 23.5% in the test set, (averaged from cross validation) which is far above random guess (20%) but which is still not satisfactory. To improve the performance of the classifier, two different measures have been taken: *winnowing* and *boosting*.

When the number of attributes is large, it becomes harder to distinguish predictive information from chance coincidences. *Winnowing* overcomes this problem by investigating the usefulness of all attributes before any classifier is constructed. Attributes found to be irrelevant or harmful to predictive performance are disregarded ("winnowed") and only the remaining attributes are used to construct the trees. As the relevance of features is one of the things we experiment on, winnowing was used. The package had a winnowing option which was used. This also makes building the trees a lot faster.

The result of this part of the experiment is as shown below

Decision tree:

semicolon $\leq$ 6.72646 (8.723425): charles (18/2)

semicolon $\geq$ 8.97227 (8.723425): :...semicolon $\geq$ 16.2095 (14.64195): charlotte (11/1)

semicolon $\leq$ 14.6293 (14.64195): :...but $\geq$ 4.71113 (4.69338): jane (15/1)

but $\leq$ 4.67563 (4.69338): :...quotation mark $\leq$ 12.285 (14.7294): emily (10)

quotation mark $\geq$ 17.1738 (14.7294): william (7)

Evaluation on training data (61 cases):

Tree Size	Errors
5	4 (6.6%) $\ll$

(a)	(b)	(c)	(d)	(e)	← classified as
14				1	(a): class jane
	16				(b): class charles
1	2	7			(c): class william
			10		(d): class emily
				10	(e): class charlotte

Evaluation on test data (17 cases):

Tree Size	Errors
5	3 (17.6%) ≪

(a)	(b)	(c)	(d)	(e)	← classified as
4		1			(a): class jane
	5		1		(b): class charles
		2			(c): class william
1			1		(d): class emily
				2	(e): class charlotte

As shown, the decision tree has an accuracy of 82.4% on the 17 patterns in the test set. Also, the tree shows that ';' (semicolon) and quotation marks were relevant attributes that define style (as they survived the winnowing).

Boosting is a wrapping technique for generating and combining classifiers to give improved predictive accuracy. Boosting takes a generic learning algorithm and adjusts the distribution given to it, by removing some training dat,) based on the algorithm's behavior. The basic idea is that, as learning progresses, the booster samples the input distribution to keep the accuracy of the learner's current hypothesis near to that of random guessing. As a result, the learning process focuses on the currently hard data. The boosting option in the package causes a number of classifiers to be constructed; when a case is classified, all the classifiers are consulted before a decision is made. Boosting often gives higher predictive accuracy at the expense of increased classifier construction time. In this case, it yields the same error rate as in winnowing in the validation set, but has 100% accuracy in the training set. The result of this part of the experiment is as follows:

Decision tree that yields the best result:

semicolon $\leq$ 8.47458 (8.723425):

:...might $\geq$ 498339 (249170.4): william (2.5)

: might $\leq$ 1.86846 (249170.4):

: :...sen len stdev $\leq$ 23.1772 (24.8575): charles (14.9)

: sen len stdev $\geq$ 26.5378 (24.8575): william (2.2)

semicolon $\geq$ 8.97227 (8.723425):

:...semicolon $\geq$ 14.6546 (14.64195):

:...sen len stdev $\leq$ 10.2292 (12.68945): jane (1.9)

: sen len stan $\geq$ 15.1497 (12.68945): charlotte (10.4)

semicolon $\leq$ 14.6293 (14.64195):

:...but $\leq$ 4.67563 (4.69338):

:...mean sen len $\leq$ 23.2097 (24.9242): emily (8.2)

: mean sen len $\geq$ 26.6387 (24.9242): william (6.4)

but $\geq$ 4.71113 (4.69338):

:...this $\leq$ 2.46609 (3.4445): jane (12.6)

this $\geq$ 4.42291 (3.4445): william (1.9)

Evaluation on training data (61 cases):

Trial	Tree Size	Errors
0	7	2 (3.3%)
1	6	5 (8.2%)
2	6	15 (24.6%)
3	6	10 (16.4%)
4	8	4 (6.6%)
5	7	8 (13.1%)
6	6	6 (9.8%)
7	6	10 (16.4%)
8	9	0 (0.0%)
boost		0 (0.0%) $\ll$

(a)	(b)	(c)	(d)	(e)	← classified as
15					(a): class jane
	16				(b): class charles
		10			(c): class william
			10		(d): class emily
				10	(e): class charlotte

Evaluation on test data (17 cases):

Trial	Tree Size	Errors
0	7	4 (23.5%)
1	6	3 (17.6%)
2	6	11 (64.7%)
3	6	12 (70.6%)
4	8	2 (11.8%)
5	7	5 (29.4%)
6	6	3 (17.6%)
7	6	10 (58.8%)
8	9	4 (23.5%)
boost		3 (17.6%) ≪

(a)	(b)	(c)	(d)	(e)	← classified as
4		1			(a): class jane
	5	1			(b): class charles
		2			(c): class william
1			1		(d): class emily
				2	(e): class charlotte

As shown, the decision tree has an accuracy of 82.4% on the 17 patterns in the test set.

2.3.3 Neural Networks — Experiments and Results

Part of the experiment carried out by Ramyaa and his team [7] included comparing the accuracy measures of between decision trees and neural networks in their use in stylometry.

Neural networks are powerful pattern matching tools. They are basically very complex non-linear modeling equations. They are especially good in situations

where the "concept" to be learned is very difficult to express as a well-defined, simple formulation, but rather is a complex, highly interrelated function of the inputs which is usually not easily transparent. This feature makes them ideal to learn a concept like "style" which is inherently intangible. The network takes all input attributes into consideration simultaneously though some are more heavily weighted than others. This differs from decision tree construction in that the style of an author is taken to be the joint product of many different features. Artificial Neural Networks has the ability to invent new features that are not explicit in the input, yet it also has the drawback that its inductive rules are inaccessible to humans.

Neuroshell – the commercial software package created by the Ward Systems Group, Inc. which is a package that creates and runs various types of neural networks, was used in the experiment by Ramyaa and his team.

Many structures of the multilayer network were experimented with before they came up with their best network. Kohonen Self Organizing Maps, probability networks, and networks based on statistical models were some of the architectures tried. Backpropogation feed forward networks yield the best result with the following architecture: 21 input nodes, 15 nodes on the first hidden layer, 11 nodes on the second hidden layer, and 10 output nodes (to act as error correcting codes). Two output nodes are allotted to a single author. (This increases the Hamming distance between the classifications - the bit string that is output with each bit corresponding to one author in the classification- of any two authors, thus decreasing the possibility of misclassification) 30% of the 61 training samples are used in the validation set which determines whether overfitting has occurred and when to stop training. The remaining 17 samples were used for testing. This testing is also used to decide the architecture of the network.

After that, on the chosen network, cross validation was done by using different test sets. There was not much variation in the results obtained. On an average, the accuracy (measured on the test set) is 88.2%. There is one misclassification Pride and Prejudice which is misclassified as written by Charlotte Brontë; there is one misclassification in Tale of Two Cities - misclassified as written by William Thackeray. These are the misclassifications that were present in all the partitions used for cross validation (irrespective of whether the patterns are on testing set or in training/validation set).

The following table gives the classification given by the network of the training and the validation sets

(a)	(b)	(c)	(d)	(e)	← classified as
11					(a): class jane
	9				(b): class charles
		10			(c): class william
			16		(d): class emily
1				14	(e): class charlotte

The following table gives the classification given by the network of the test set

(a)	(b)	(c)	(d)	(e)	← classified as
2					(a): class jane
	2				(b): class charles
		2			(c): class william
		2	4		(d): class emily
				5	(e): class charlotte

Also, the patterns that decision trees misclassify are not the same as those misclassified by the neural network, although the misclassification is persistent within on learning technique i.e. for more than one architecture, the same patterns get misclassified in neural network; Decision trees, with different parameters also have trouble with a same set of patterns (different from the set that troubles Artificial Neural Networks). The Neural Network is obviously looking at some different attributes from the ones that the decision trees look at.

2.3.4 Conclusion

Both Decision Trees and Artificial Neural Networks yield a significantly higher accuracy rate than random guess, which shows that the assumption that there is a quantifiable unconscious aspect in an author's writing style is well justified. Ramyaa[7] achieved an accuracy of 82.4% on the test set using decision trees, and an accuracy of 88.2% on the test set using neural networks

Although the neural networks yielded a better numerical performance, and are considered inherently suitable to capture an intangible concept like style, the decision trees are human readable making it possible to define style. Also, the results obtained by both methods are comparable [7].

Decision trees are rule based, explicit learners. Due to their rule-based nature, it is easy to read and understand a decision tree. Thus, it is possible to "see the style"

in the texts with these trees. Also, the results indicate that the unconventional features we used are quite useful in classification of the authors [9].

It is for these reasons why the Decision Trees were chosen for the implementation of the proposed project, as opposed to Neural Networks.

2.4 Examples of Applications of Stylometry in the Real World

Fraud detection, email classification, deciding the authorship of famous documents, attributing authors to pieces of texts in collaborative writing, and software forensics are some of the many uses of authorship attribution [7].

In some criminal, civil, and security matters, language can be evidence. A suicide note, a threatening letter, anonymous communications, business emails, blog posts, trademarks—all of these can help investigators, attorneys, human resource executives and private individuals understand the heart of an incident. When you are faced with a suspicious document, whether you need to know who wrote it, or if it is a real threat or a real suicide note, or if it is too close for comfort to some other document, you need reliable, validated methods. [14]

Today with the help of many other significant events in stylometry, its techniques are being widely applied in various areas, such as, disease detection, email authentification, forensic science and court trials.

Electronic communication is one of the popular ways of communication in this era. E-mail communication is the most popular way of electronic communication. Internet works as the backbone for these communications. [10]. Computer viruses and worms are now commonly distributed by email, and this brings up the need to identify the authors of messages through non repudiation. Calix and his team, prior Pace Computer Science students [12], created a program in C# to solve this problem. The program has a Graphical User Interface so as to simplify the task of dtermining authorship by automating the identification process. Users help the program to recognize authors by feeding it with emails from unknown others for it to learn their linguistic style. Subsequently, the users feed a a set of sample e-mails by unknown authors for comparison. The program considers 55 stylistic features.

Examples of using stylometric techniques as a forensic tool include the appeal in London of Tommy McCrossen in July 1991 and the pardon for Nicky Kelly from the Irish government in April 1992 [7]. In the McCrossen case, stylometric evidence

which cast doubt over the authenticity of the defendant's confession played a vital role in persuading the appeal judges that he had been wrongly convicted.

Similarities with private letters helped to identify the style of the Unabomber's manifesto. Theodore Kaczynski perpetrated a number of bomb attacks on universities and airlines between 1978 and 1995, under the alias Unabomber. He promised to stop if his 35,000-word anti-industrialist "manifesto" was published in major newspapers. Distinctive writing style and turns of phrase enabled him to be identified [15].

In an experiment conducted by Lakshmi [10], he and his team were able to prove in their paper that stylometric techniques can do more than just authorship attribution of documents. The proved that Stylometry can also identify the gender of the human.

The JGAAP (Java Graphical Authorship Attribu- tion Program) system is a Java based stylometric tool that runs on both Windows and Linux operating systems. Using standard Java 1.6 technology, a clear theoreti- cal framework, and an easy, user-extensible interface, JGAAP provides for objective testing of proposed methods. Specifically, the JGAAP framework takes "Documents," essentially strings generated from files of interest by any class instantiating the proper interface, converts them to "EventSets," again defined as any class implementing the proper interface, then analyzes them. Currently available EventSets include characters, words, syllables, reaction times, n-grams of any of the above, or "most common" versions of any of the above [13].

Stylometric analysis was used to settle the dispute of the Federalist Papers [5]. The Federalist Papers were written during the years 1787 and 1788 by Alexander Hamilton, John Jay, and James Madison. These 85 'propaganda' tracts were intended to help the U.S. constitution ratified. They were all published anonymously under the name 'Publius'. The authorship of the Federalist Papers was disputed from the beginning. Both Hamilton and Madison produced lists that claimed some of the same papers.

Pioneering stylometric methods were famously used by Mosteller and Wallace in 1964 to attempt to answer the question as to who the authorship of the papers was attributed to. This dispute is now considered settled.

2.5 Conclusion

With the literature review of the system done, the researcher hereby proceeds to the system analysis phase of the project.

3 SYSTEM ANALYSIS

3.1 Introduction

This chapter deals with the analysis of the proposed system, and the requirements that it needs, both for its development and its running. The requirements comprise of:

1. Functional Requirements
2. Non-Functional Requirements
3. Software Requirements
4. Hardware Requiremets

Also included in this chapter is the feasibility study of the proposed project.

3.2 Overview

This chapter begins with the functional requirements. Then what follows are the non-functional requirements. After that follows the system requirements. Finally, the chapter ends with the feasibility study of the proposed project.

Below are the functional and non functional requirements of the system.

3.2.1 Functional Requirements

Functional requirements specify what a system should do. They specify a function or component that the system must be able to perform.

1. Admin can login and logout.
2. Admin can register a new student.
3. Admin can register a new source.
4. The system can accept an essay, split it into sections, extract data (linguistic stylistic features) from each section and use it to generate a decision tree.
5. The system can use the decision tree generated above to compare the linguistic styles of a subsequent essay of the student.

6. The system can provide results of the stylometric analysis of the students' essays.

3.2.2 Non-Functional Requirements

Non-functional requirements refer to the requirements of a system that describe its characteristics, as opposed to its functionalities. It is a statement of how a system must behave, it is a constraint upon the system's behavior.

1. REST protocol which provides flexible service based architecture. This is desirable for future extensions.
2. 24 hours per day availability.
3. A good component design for easier comprehension and better performance.
4. A user friendly user interface for easier and convenient user experience.

3.3 System Requirements

Hardware Requirements	**Software Requirements Size**
RAM : 256MB	OS : Windows, Linux, Mac
Internal HD : 10GB minimum	Front-end : HTML, CSS, JS
External HD	Back-end : Ruby on Rails
Screen : 11 inch minimum	Database : PostgreSQL

3.4 Feasibility Study

This refers to the study to determine the extent to which a project can be performed successfully. It determines whether the solution considered to accomplish the requirements is practical and workable in the software or not.

3.4.1 Technical Feasibility

1. The programming language chosed to code the proposed project has been in existence for a couple of years. It is reliable and stable.
2. The proposed project makes use of the ID3 algorithm which is readily available in many packages spanning across different programming languages. Examples include the AI4R and DecisionTree gems in Ruby on Rails and

the J8 library in Weka package. These all implement the Quinlan C4.5 ID3 algorithm.

3.4.2 Operational Feasibility

The cost to be incurred in the development of the proposed project fall well within the range of the allocated KES 7,000. The cost consists if the sum of charges from :

1. hardware, for instance flashdisk, mouse and modem
2. purchase of internet data bundles

4 SYSTEM DESIGN

4.1 Introduction

This software design document describes the architecture and system design of a pattern recognition system that identifies the linguistic styles in essays for the sole purpose of authorship attribution. The system uses ID3, an algorithm in Decision Trees, to achieve this.

4.2 Scope

The scope of this project includes development of a web application for determining the authorship integrity of essays using the ID3 algorithm in Decision Trees.

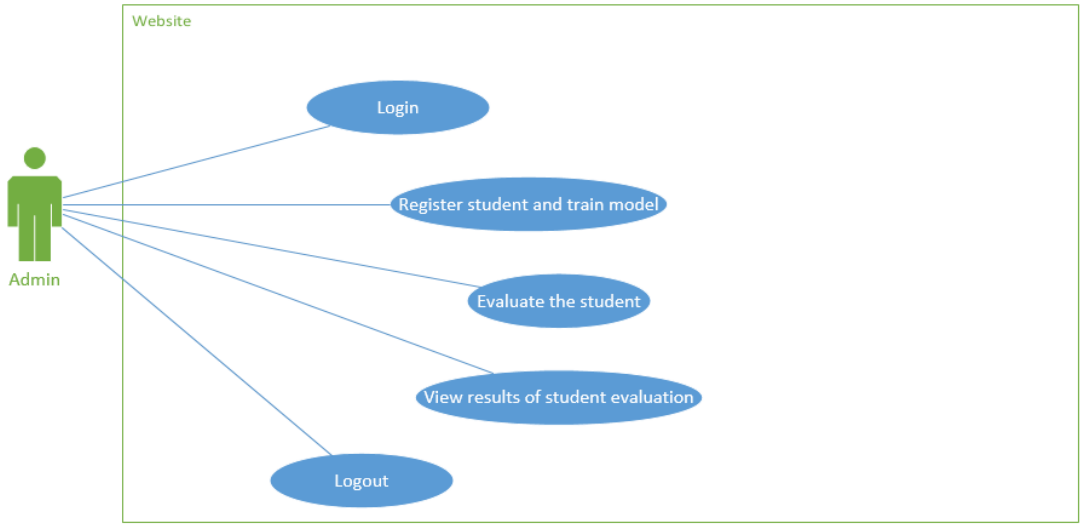
4.3 Overview

This chapter is organized as follows. It starts with an introduction then it proceeds to system use case description section which explains an abstract view of the interaction between the system and its users. Then follows a subsection on the system's component design. Following this is the structural modelling section which includes subsections on the system's data dictionary and the the system's ERD relations.

4.4 Use Case Descriptions

The use case communicates at a high level what the system needs to do. Use cases capture the typical interaction of the system with the system's users (end users and other systems). The following is the use case of the proposed system.

Figure 1: Use case diagram showing the interaction between the admin and the system



4.5 Component Design

The system is divided into the following five modules:

1. *Data module module :*

   This module is responsible for handling all input data used in the system. This comprises of details of the users and their essays. All data in the system is handled by this module.

2. *Essay preprocessing module :*

   This module preprocesses the input data before it is fed to the *Stylometric analysis module*. It is responsible for putting data into a format which can be understood by the Stylometric analysis engine.

3. *Stylometric analysis module :*

   This module is the core functionality of the system. It is responsible for the processing and analysis of the users' essays, and compares their results. This is the module that is responsible for authorship attribution.

   This module consists primarily of the J48 function which is an implementation of the Quinlan C4.5 ID3 algorithm.

   At this module, the training csv/arff files will be used to train the model and generate a resulting decision tree. The model will learn the linguistic styles of the given author, after which it will take in the testing csv/arff file and classify it using the generated decision tree.

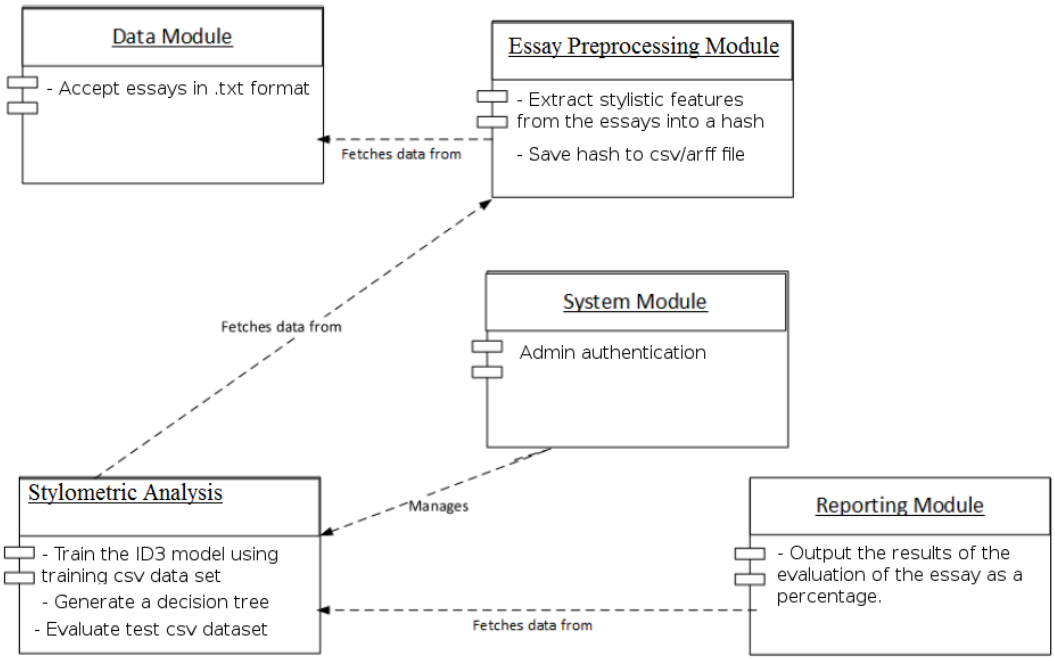
4. *Reporting module :*

   This module is used to format and display output from the *Stylometric analysis engine*. The output will be diaplayed as a percentage of the linguistic similarities of the essay being evaluated, and a previous one that was used to train the system to learn the stylometric style of a particular user.

5. *System module :*

   This module handles the all operations of the system which are not required for the stylometric analysis, but the system needs them, for instance, management and user authentication.

Figure 2: Component Diagram showing the modules that comprise the system



4.6 Structural Modelling

A structural, or conceptual, model describes the structure of the data that supports the business processes in an organization. The structural model presents the logical organization of data without indicating how the data are stored, created, or manipulated so that analysts can focus on the business without being distracted by technical details.

4.6.1 Data Dictionary

Below is the **Student table**, **Admin table** and **Source table**with corresponding datatypes and descriptions.

4.6.2 ERD

The diagram that follows shows the structural model of the proposed system using the Crowfoot's notation.

Table 3: Student Table

Field	Datatype	Description
student_id	integer	Stores primary key
name	string	Stores the name of the student
essay	text	Stores the initial text of the student
source_id	integer	Foreign key storing the primary key of the source
date	datetime	A date object storing the date for the data

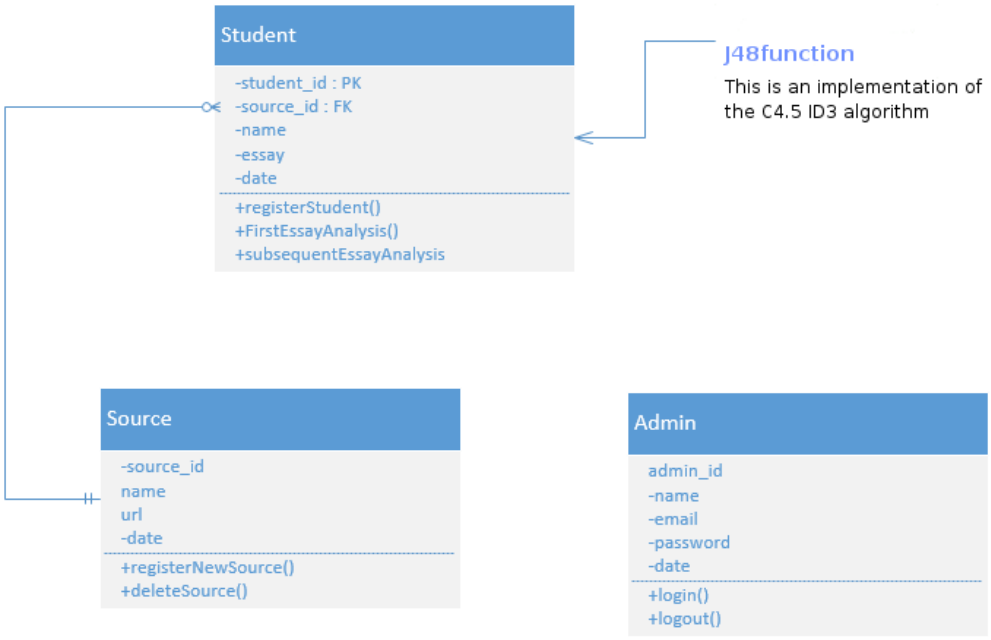
Table 4: Admin Table

Field	Datatype	Description
admin_id	integer	Stores primary key
name	string	Stores the name of the administrator of the system
email	string	Stores the email of the administrator of the sustem
password	text	A string storing the administrator's password
date	datetime	A date object storing the date for the data

Table 5: Source Table

Field	Datatype	Description
source_id	integer	Stores primary key
name	text	A string storing the name of the source of data
url	text	A string storing the website url for the source of data
date	datetime	A date object storing the date for the data

Figure 3: Entity-Realtionship Diagram showing the interactiobetween the users and the system



4.7 Human Interface Design

4.7.1 Overview of Human Interface Design

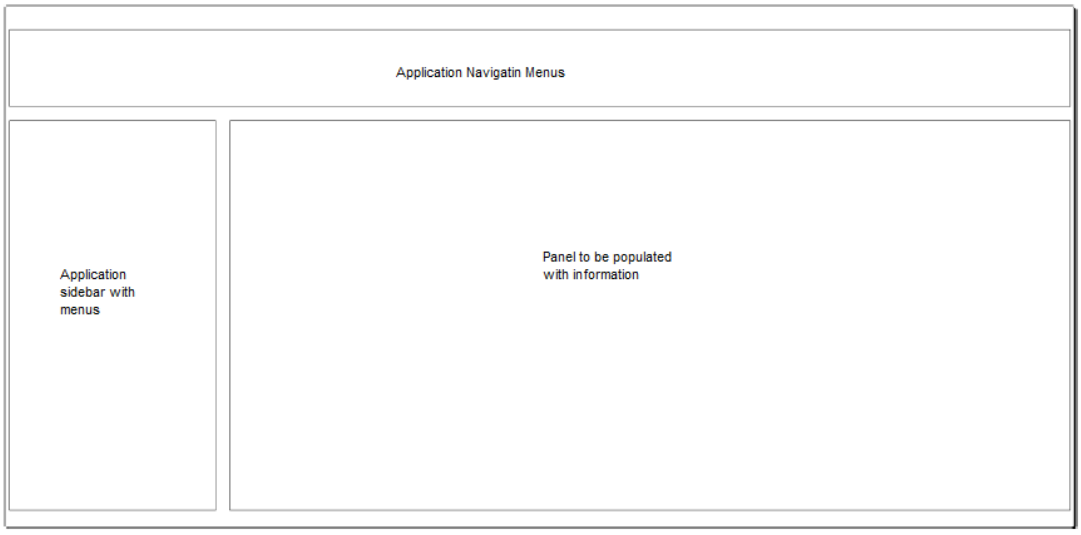
An administrator of the system will be the one in charge of registration of both students and the sources of data. He/She will also be the one to trigger the stylometric analysis of the essay The administrator will be presented with a login screen and on login he will be taken to the homepage from where he/she is presented with options to register sources and students, and to evaluate a student's essay.

4.7.2 Screen Images

This section displays screen shots showing the interface from the user's perspective.

Figure 4: A wireframe of the login window of the system

Figure 5: A wireframe of the home page window of the system



5 SYSTEM IMPLEMENTATION

5.1 Introduction

The proposed system described in the previous chapter (System Design) is implemented using Ruby programming language; more specifically, Ruby on Rails, which is an Object-Oriented framework of the Ruby programming language.

This chapter has Ruby code snippets from the system and their explanation, and the screenshots of the various windows during the running of the system. The system itself is a web application that uses stylometric analysis to chieve authorship attribution of essays. The application outputs the probability that the claimant of an essay the true author of it.

The administrator of this web application, upon logging in, will be able to enter register a student, including an essay that the student has written. This will be the essay that will be used to train a decision tree model and use it to test subsequents essays of the student.

The output of training the J48 decision tree model, and the results of the stylometric analysis will be displayed accordingly on a webpage of the web application.

The system is divided into the following modules:

1. *Data module :*

   This module is responsible for handling all input data used in the system. This comprises of details of the users and their essays. All data in the system is handled by this module. The system will accept essays in .txt format. The input essays will constitute the training and/or testing datasets for the system.

2. *Essay pre-processing module :*

   This module preprocesses the input data before it is fed to the Stylometric analysis module. It is responsible for putting data into a format which can be understood by the Stylometric analysis engine. At this module, the specified stylistic features are used to extract data from the input essays and store in a hash function. Examples of the stylistic features used are relative frequencies of prepositions (on, for, in, above, over), relative frequencies of pronouns (it, she, it, this, I, you), relative frequencies of conjuctions (and, but, yet, if), relative frequencies of contractions ('ve, 'll, 're, 'nt), and the relative frequencies of punctuation marks (.,!?""—/-). These features will specified

in a file called stopwords.txt in the system. The hash function is then stored in either a CSV (Comma Separated Values) or ARFF (Attribute Relation File Format) files.

3. *Stylometric analysis engine :*

   This module is the core functionality of the system. It is responsible for the processing and analysis of the users' essays, and compares their results. This is the module that is responsible for authorship attribution. This module consists primarily of the J48 function which is an implementa- tion of the Quinlan C4.5 ID3 algorithm. At this module, the training csv/arff files will be used to train the model and generate a resulting decision tree. The model will learn the linguistic styles of the given author, after which it will take in the testing csv/arff file and classify it using the generated decision tree.

   The C4.5 ID3 algorithm use pruning. Pruning is a way of reducing the size of the decision tree. This will reduce the accuracy on the training data, but (in general) increase the accuracy on unseen data. It is used to mitigate overfitting , where you would achieve perfect accuracy on training data, but the model (the J48 decision tree) you learn is so specific that it does not apply to anything but that training data.

   The C4.5 ID3 algorithm also uses rulesets. Ruleset is a class that trains an ID3 Tree with 2/3 of the training data, converts it into a set of rules and prunes the rules with the remaining 1/3 of the training data.

4. *Reporting module :*

   This module is used to format and display output from the Stylometric analysis engine. The output will be diaplayed as a percentage of the linguistic similarities of the essay being evaluated, and a previous one that was used to train the system to learn the stylometric style of that particular student.

5. *System module :*

   This module handles all the operations of the system which are not required for the stylometric analysis, but the system needs them, for instance, management and user authentication. The system provides a login authentication upon loading the url of the web application.

5.2 Component Implementation

In this section, the researcher will summarize implementation of each of the components of the system, give code snippets, as well as explain them.

5.2.1 Data Component

This component handles all the input data used in the system. The input data includes the details of the student and the initial essay, the details of the source, and the essay to be evaluated.

student_params() method

```
def student_params
  params.require(:student).permit(:source_id , :name, :essay)
end

def set_student
  @student = Student.find(params[:id])
  session[:student_id] = @student.id
end
```

source_params() method

```
def set_source
  @source = Source.find(params[:id])
end

def source_params
  #code
  params.require(:source).permit(:name, :url)
end
```

test_params() method

```
def test_params
  params.require(:test).permit(:evaluate)
end

def load_student
      @student = Student.find(params[:student_id])
end
```

5.2.2 Essay Pre-processing Module

This component handles the preprocessing of the initial essay and the essay to be evaluated. The preprocessing of the essays involve using predetermined stylometric features to extract data from them and store them in a hash. The hash is then stored to an external csv file.

Below is a summary of the core methods found in this component:

process_initial_essay(file) method

```
def process_initial_essay(my_file, path_to_save_csv)

    # Initialize a hash with a default of 0
    @countedWords = Hash.new(0)
    @section = Array.new
    @section_length = Array.new
    @counts = Array.new
    @classifierRelativeFrequency = Array.new

    # code to read the essay file and convert it to lowercase
    my_file = my_file.read.downcase

    words = my_file.scan(/\w[\w']*/) #now catches contractions

    # Count words (keys) and increment their value
    words.each {|word| @countedWords[word] += 1 }

    # Count the number of words in the essay
    @wordCount = words.size

    # Divide the essay into groups
    full_section = words.each_slice(GROUP_SIZE).to_a

    # Number of groups
    @number_of_groups = (@wordCount / GROUP_SIZE).ceil

    # Read classifiers from the classifier csv file
    file = (File.read(Rails.root.join('app','models','concerns','stopwor
    @csv_headers = file.split(" ")
```

```
  # Create a hash to store the relative frequencies of the classifiers
  1.upto(@number_of_groups) { |x|
    @classifierRelativeFrequency[x-1] = Hash.new(0)
  }

  # Open MainDataSet a csv file and add the classifiers as headers
  CSV.open(path_to_save_csv, "a+") do |csv|

    # Loop to assign sections and get their sizes
    1.upto(@number_of_groups) { |x|

      # Assign the sections of the full section to variable arrays
      @section[x-1] = full_section[x-1]

      # Get the length of the sections and store them in an array
      @section_length[x - 1] = @section[x - 1].size

      # Create a hash for statistical analysis
      @counts[x-1] = Hash.new(0)

      # Add words to hash and increments count
      @section[x - 1].each {|word|
        @counts[x - 1][word] += 1
      }

      # Calculate the relative frequency of the features in each sectio
      1.upto(@csv_headers.size - 1) { |c|
        @classifierRelativeFrequency[x-1]["#{@csv_headers[c-1]}"] = (1
      }

      # Add the id of the author as the value of the class
      @classifierRelativeFrequency[x-1].merge!("class" => @student.id)

      # Add the relative frequencies to the csv file
      csv << @classifierRelativeFrequency[x-1].values.to_a
    }
  end
end

GROUP_SIZE = 90.0
```

process_test_essay(file) method

```
def process_test_essay(my_file, path_to_save_csv)

    # Initialize a hash with a default of 0
    @countedWords = Hash.new(0)
    @section = Array.new
    @section_length = Array.new
    @counts = Array.new
    @classifierRelativeFrequency = Array.new

    # code to read the essay file and convert it to lowercase
    my_file = my_file.read.downcase

    words = my_file.scan(/\w[\w']*/) #now catches contractions

    # Count words (keys) and increment their value
    words.each {|word| @countedWords[word] += 1 }

    # Count the number of words in the essay
    @wordCount = words.size

    session[:wordCount] = @wordCount

    # Divide the essay into groups
    full_section = words.each_slice(GROUP_SIZE).to_a

    # Number of groups
    @number_of_groups = (@wordCount / GROUP_SIZE).ceil

    session[:numberOfGroups] = @number_of_groups

    # Read classifiers from the classifier csv file
    file = (File.read(Rails.root.join('app','models','concerns','stopwor
    @csv_headers_modified = file
    @csv_headers_modified = @csv_headers_modified.gsub("'","")
    # Put the classifiers in an array
    @csv_headers = file.split(" ")
    @csv_headers_modified = @csv_headers_modified.split(" ")
```

```
# Create a hash to store the relative frequencies of the classifiers
1.upto(@number_of_groups) { |x|
  @classifierRelativeFrequency[x-1] = Hash.new(0)
}

# Open  test csv file and add the classifiers as headers
CSV.open(path_to_save_csv, "wb") do |csv|

  csv << @csv_headers_modified

  # Loop to assign sections and get their sizes
  1.upto(@number_of_groups) { |x|

    # Assign the sections of the full section to variable arrays
    # starting from index 0
    @section[x-1] = full_section[x-1]

    # Get the length of the sections and store them in an array
    # starting from index 0
    @section_length[x - 1] = @section[x - 1].size

    # Create a hash for statistical analysis
    @counts[x-1] = Hash.new(0)

    # Add words to hash and increments count
    @section[x - 1].each {|word|
      @counts[x - 1][word] += 1
    }

    # Calculate the relative frequency of the features in each sectio
    1.upto(@csv_headers.size - 1) { |c|
      @classifierRelativeFrequency[x-1]["#{@csv_headers[c-1]}"] = (1
    }

    # Add the id of the author as the value of the class
    @classifierRelativeFrequency[x-1].merge!("class" => @student.id)

    # Add the relative frequencies to the csv file
    csv << @classifierRelativeFrequency[x-1].values.to_a
  }
```

```
    end
  end
  # set group size to be 90
  GROUP_SIZE = 90.0
```

5.2.3 Stylometric Analysis Engine

This is the module that is responsible for stylometric analysis and authorship attribution. It consists primarily of the J48 function which is an implementa- tion of the Quinlan C4.5 ID3 algorithm. At this module, the training csv file is be used to train the model and generate a resulting model decision tree. The model learns and classifies the authors (students) based on their linguistic styles, after which it will take in the testing csv file to be evaluated and classify it using the generated decision tree.

evaluateModel() method

```
def evaluateModel

    @startedEvaluation = 1
    session[:startedEvaluation] = @startedEvaluation

    loadEnvironment()

    #load the training data
    path = Rails.root.join('app','models','csv_files',"MainDataSet.csv")
    train_src = Rjb::import("java.io.File").new(path)
    train_csvloader = Rjb::import("weka.core.converters.CSVLoader").new
    train_csvloader.setFile(train_src)
    train_data = train_csvloader.getDataSet

    #load test data
    path = Rails.root.join('app','models','csv_files',"test.csv").to_s
    test_src = Rjb::import("java.io.File").new(path)
    test_csvloader = Rjb::import("weka.core.converters.CSVLoader").new
    test_csvloader.setFile(test_src)
    test_data = test_csvloader.getDataSet

    # NumericToNominal
    ntn = Rjb::import("weka.filters.unsupervised.attribute.NumericToNomi
```

```
ntn.setInputFormat(train_data)
ntn.setInputFormat(test_data)
train_data = Rjb::import("weka.filters.Filter").useFilter(train_data
test_data = Rjb::import("weka.filters.Filter").useFilter(test_data,

#create a J48 model
tree = Rjb::import("weka.classifiers.trees.J48").new

train_data.setClassIndex(train_data.numAttributes() - 1)
test_data.setClassIndex(test_data.numAttributes() - 1)
tree.buildClassifier train_data

#serialize model
sh = Rjb::import("weka.core.SerializationHelper")
sh.write("/tmp/weka.model", tree);

#deserialize model
sh = Rjb::import("weka.core.SerializationHelper")
tree = sh.read("/tmp/weka.model");

@dtreeString = tree.toString
puts @dtreeString

# Write out to a dot file
@student_id = session[:student_id]
@classname = train_data.classAttribute.toString.split(' ')[1] + @stu

session[:classname] = @classname

graph = tree.graph.gsub(/Decision Tree {/, "Decision Tree {\n#{@class
File.open(Rails.root.join('app','assets','images','dots', @classname
`dot -Tgif < /home/ronnie/Rails/StylometryProject/app/assets/images/

puts "Generated tree for #{@classname}"

# Classify training data
preds_train = Array.new
points_train_array = Array.new
points_train = train_data.numInstances
points_train.times do |instance|
```

```
  pred = tree.classifyInstance(train_data.instance(instance))
  point = train_data.instance(instance).toString
  point = point.split(",") << pred
  points_train_array << point
  preds_train << pred
  puts "#{point} : #{pred}"
end

preds_train = preds_train.to_s.gsub("]","")
preds_train = preds_train.to_s.gsub("[","")
preds_train = preds_train.to_s.gsub(" ","")
preds_train = preds_train.split(",")
#puts hash_function(preds_train)

puts "Finished classifying train points"

# Classify test data
preds_array = Array.new
test_data.numInstances.times do |instance|
        pred = tree.classifyInstance(test_data.instance(instance))
        preds_array << pred
        puts "#{pred}"
end

puts "Finished prediction"

puts preds_array

preds_array = preds_array.to_s.gsub("]","")
preds_array = preds_array.to_s.gsub("[","")
preds_array = preds_array.to_s.gsub(" ","")
preds_array = preds_array.split(",")

session[:totalTestInstances] = preds_array.size

hash1 = hash_function(preds_array)
puts "Hash 1 : #{hash1}"

# Populate an array with the id number of the students/authors
arrayForIDS = Array.new
```

```
0.upto(hash1.size - 1) {|j|
  0.upto(points_train_array.size - 1) {|i|
    if hash1.keys[j].to_i == points_train_array[i].last
      if arrayForIDS.include? points_train_array[i][-2]
        # Do nothing
      elsif points_train_array[i][-2].to_i <= Student.all.count
        arrayForIDS << points_train_array[i][-2]
      end
    end
  }
}

# Create hash with key as student id and value as frequency
idFreq = Hash.new(0)
keys = arrayForIDS
0.upto(arrayForIDS.size - 1){|i|
  idFreq[keys[i]] = hash1.values[i]
}

puts "Array for IDS : #{arrayForIDS}"
puts "ID Frequency : #{idFreq}"

id_number = Array.new
@name = Array.new

corrects = Array.new
@percentages = Array.new
0.upto(idFreq.length - 1){|i|
  id_number[i] = idFreq.keys[i].to_i
  @name[i] = Student.find(id_number[i]).name.to_s
  corrects[i] = idFreq[keys[i]]
  if corrects[i].nil?
    corrects[i] = 0.0
  end
  puts corrects[i]
  @percentages[i] = (100 * corrects[i].to_f / preds_array.size.to_f)
}

session[:name] = @name
session[:percentages] = @percentages
```

```
    # Calculate percentage
    studentid = session[:student_id].to_s
    @correct = idFreq[studentid].to_i
    @wrong = preds_array.size - @correct
    @percentage = (100 * @correct.to_f / preds_array.size).round(4)

    session[:correctlyClassified] = @correct
    session[:wronglyClassified] = @wrong
    session[:percentage] = @percentage

    redirect_to request.referer
  end
```

5.2.4 Reporting Module Component

Reporting module is used to format and display output from the *Stylometric analysis engine* in the required format using percentages showing the probability of the essay being evaluated belonging to claimant author (student). This module defines how the interface of various other modules look as far as output from the system is concerned.

Graphing the system output

```
<div class="col-md-6 skin-white">
      Name : <%= @student.name %> <br />
      ID : <%= @student.id %> <br />
      Source : <%= @student.source.name %> <br />
      Created on : <%= @student.created_at.strftime("%d/%m/%Y at %I:%M%p

      <% if @startedEvaluation == 1 %>
        <!--What to display after evaluation-->
        Size of essay : <%= @wordCount %> <br />
        Number of Sections : <%= @number_of_groups %> <br />

        <%= image_tag("gifs/#{@classname}.gif", :class=>"img-responsive"
        <%= @dtreeString %> <br />

        Total test instances : <%= @totalTestInstances %> <br />
        Correctly classified : <%= @correct %> <br />
```

```
        Wrongly classified : <%= @wrong %> <br />
        Percentage : <%= @percentage %>% probability that it was written

    <% else %>
      <!--What to display before beginning of evaluation-->
    <% end %>
    <hr>
</div>

<div class="col-md-3 skin-white">
  <% if @startedEvaluation == 1 %>
    <% if @percentage >= 75.0 %>
      <h2 id="good"><%= @percentage %>%</h2>
    <%elsif @percentage >= 50.0 %>
      <h2 id="warning"><%= @percentage %>%</h2>
    <% else %>
      <h2 id="bad"><%= @percentage %>%</h2>
    <% end %>

    <% 0.upto(@names.length - 1) do |i| %>
      <%= @names[i] %> : <%= @percentages[i] %>% <br>
    <% end %>

  <% else %>

  <% end %>
</div>
```

5.2.5 System Module Component

This component is responsible for admin management of the system. The system provides a login authentication upon loading the url of the web application.

Requirement for authentication

```
before_filter  :authenticate_admin!
```

CRUD (Create, Read, Update and Delete) methods for student

```
before_filter  :authenticate_admin!
  before_action :set_student, only: [:show, :edit, :update, :destroy]

  def index
    @students = Student.all
    set_sessions_to_nil()
  end

  def new
    @source_options = Source.all.map{|s| [s.name, s.id]}
    @student = Student.new
    set_sessions_to_nil()
  end

  def show
    #@dtreeString = session[:dtreeString]
    @classname = session[:classname]
    @wordCount = session[:wordCount]
    @number_of_groups = session[:numberOfGroups]
    @correct = session[:correctlyClassified]
    @wrong = session[:wronglyClassified]
    @percentages = Array.new
    @percentages = session[:percentages]
    @percentage = session[:percentage]
    @totalTestInstances = session[:totalTestInstances]
    @names = Array.new
    @names = session[:name]
    @id = session[:student_id]

    @startedEvaluation = session[:startedEvaluation]

  end

  def create
    @student = Student.new(student_params)

    respond_to do |format|
      if @student.save
        format.html{redirect_to @student, notice: 'Student was successfu
        format.json{render :show, status: :created, location: @student}
```

```
      my_file = @student.essay
      path_to_save_csv = Rails.root.join('app','models','csv_files', "
      process_initial_essay(my_file, path_to_save_csv)
    else
      format.html{render :new}
      format.json{render json: @student.errors, status: :unprocessable
    end
  end
end

def edit
  @source_options = Source.all.map{|s| [s.name, s.id]}
  set_sessions_to_nil
end

def update
  @student = Student.new(student_params)
  respond_to do |format|
    if @student.save
      #code
      format.html{redirect_to @student, notice: 'Student was successfu
      format.json{render :show, status: :created, location: @student}
    else
      format.html{render :new}
      format.json{render json: @student.errors, status: :unprocessable
    end
  end
  set_sessions_to_nil()
end

def destroy
  #code
  @student.destroy
  respond_to do |format|
    format.html{redirect_to students_url, notice: 'Student was success
    format.json{head :no_content}
  end
end
```

5.3 Human Interface Implementation

5.3.1 Overview of User Interface

Upon entering the url of the application and loading the system, the admin will be presented with a login screen. On successful authentication, the admin will be taken to the home page where he will be presented with a menu list for the functionality of the system. From the menu, the admin can choose to register a new source, register a new student, or evaluate an existing student.

The "register source" and "register student" options take the admin to a new page where he/she fills out a form, giving details of the respective entities, and submits it.

The "evaluate student" menu takes the admin to a a page containing a list of all registered student. In this page, he/she has to select the student to be evaluated. Upon selection, the admin is asked to input the essay to be evaluated, and presented with a button to "train the model and evaluate". Clicking this button triggers the start of the training and evaluation process, and on completion, the results are displayed in that page for the admin to see.

5.3.2 Screen Images

This section display screen shots showing the interface from the user's perspective.

Figure 6 shows a screenshot of the login window of the system.

Figure 7 shows a screenshot of the home page of the system.

Figure 8 shows a screenshot of the page for adding a new source to the web application.

Figure 9 shows a screenshot of the page for adding a new student/author to the web application.

Figure 10 shows a screenshot of the evaluation page, before an evaluation is performed.

Figure 11 shows a screenshot of the page for adding a new essay to be evaluated.

Figure 12 shows a screenshot of the evaluation page, after an evaluation has been done. The results of the stylometric analysis are displayed here.

Figure 6: Screenshot of the login window

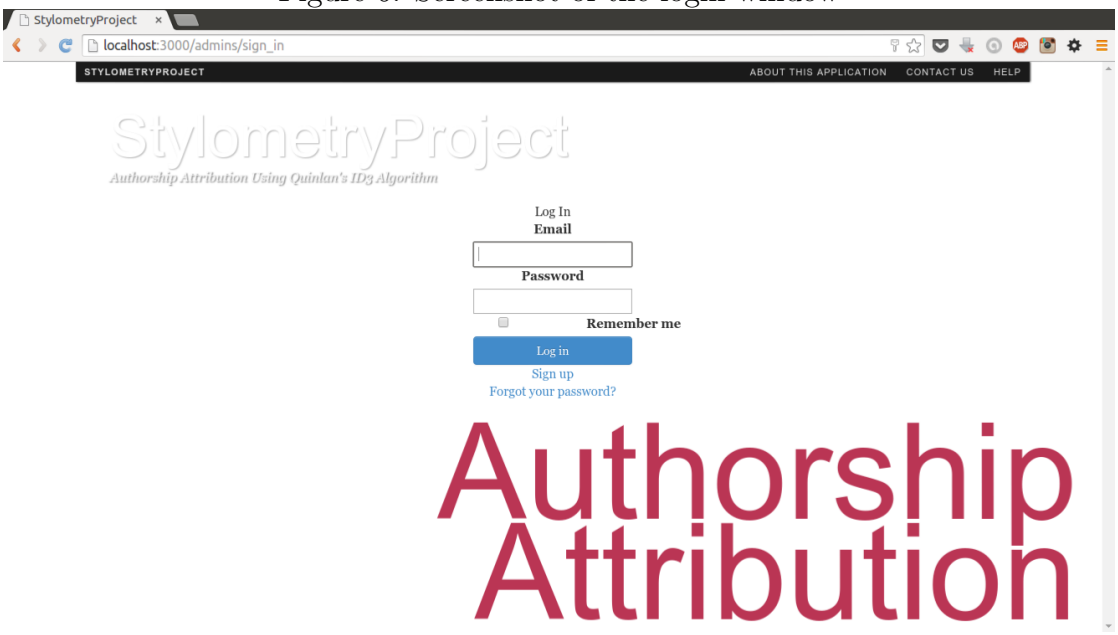


Figure 7: Screenshot of the homepage

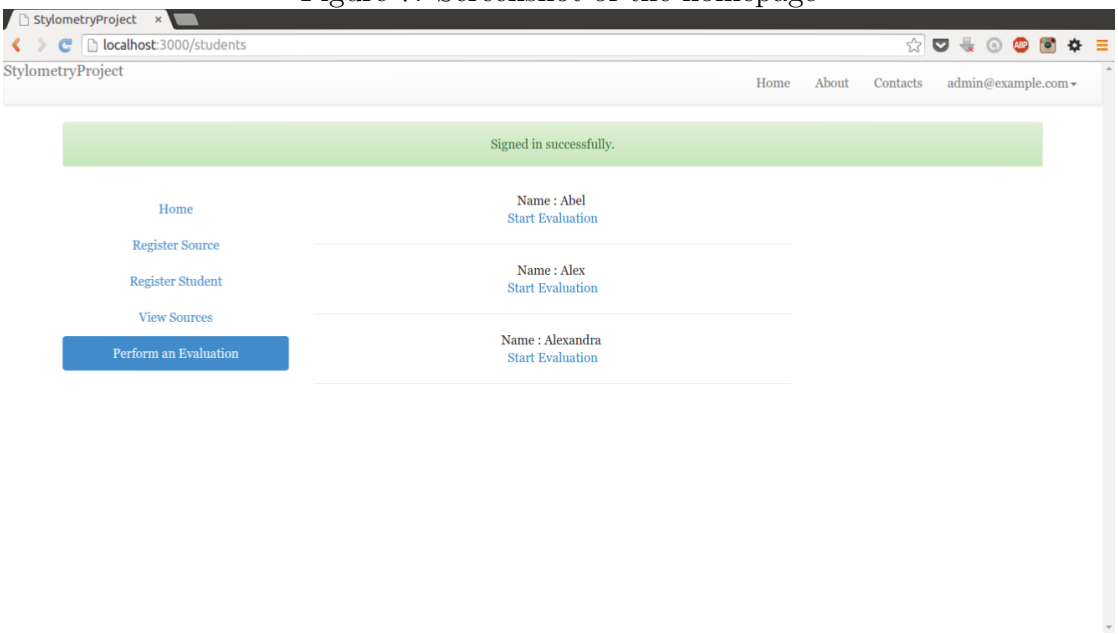


Figure 8: Screenshot of the new_source page

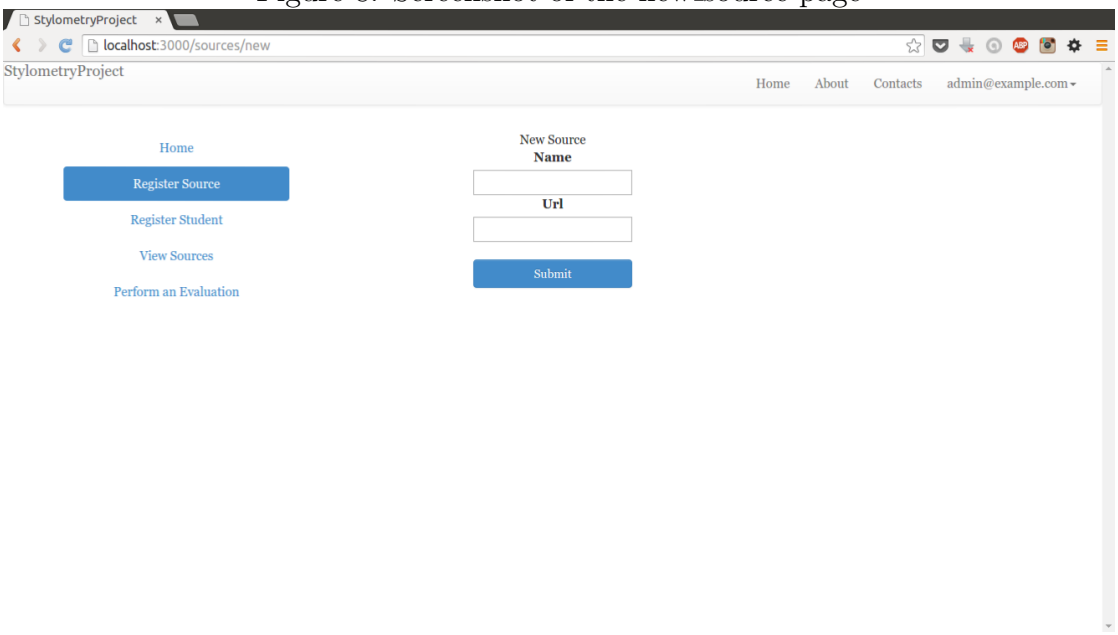


Figure 9: Screenshot of the new_student page

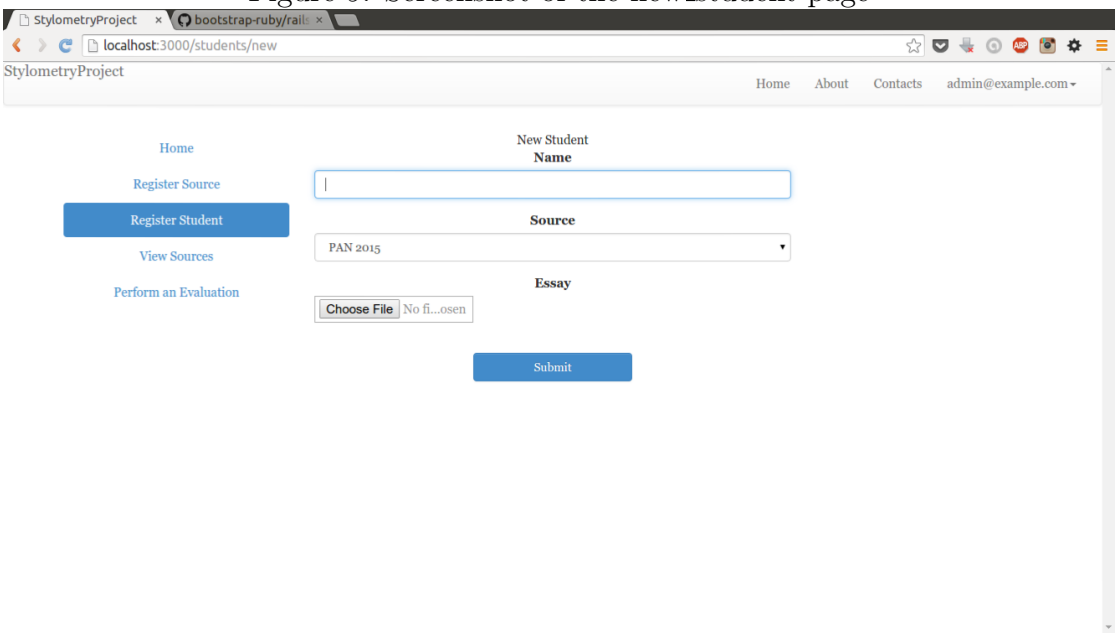


Figure 10: Screenshot of the evaluation page (before evaluation)

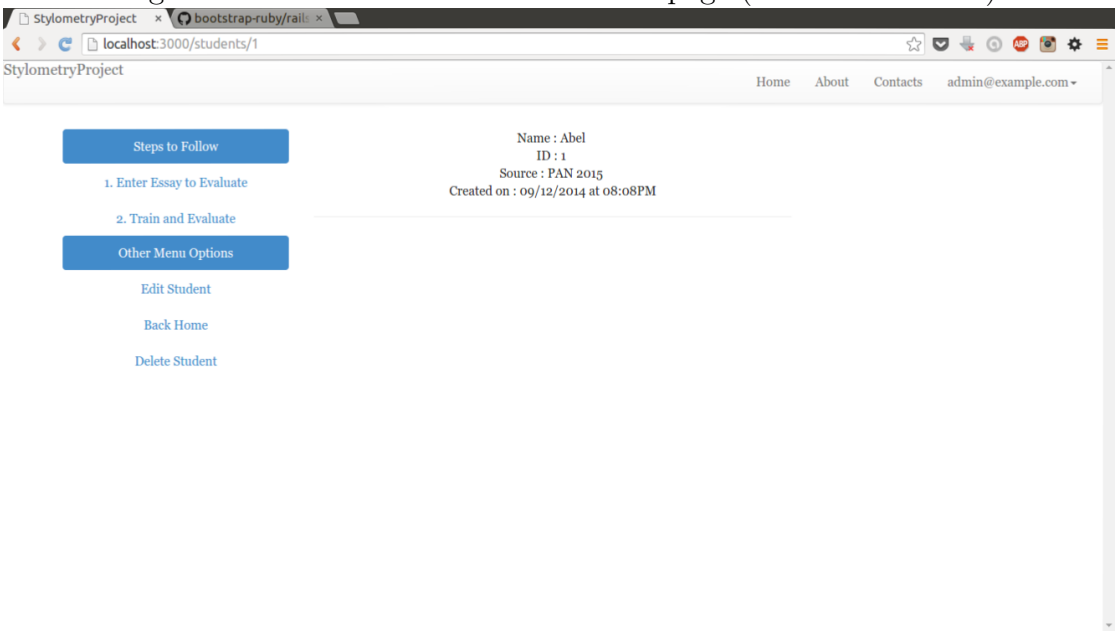


Figure 11: Screenshot of the new_test page

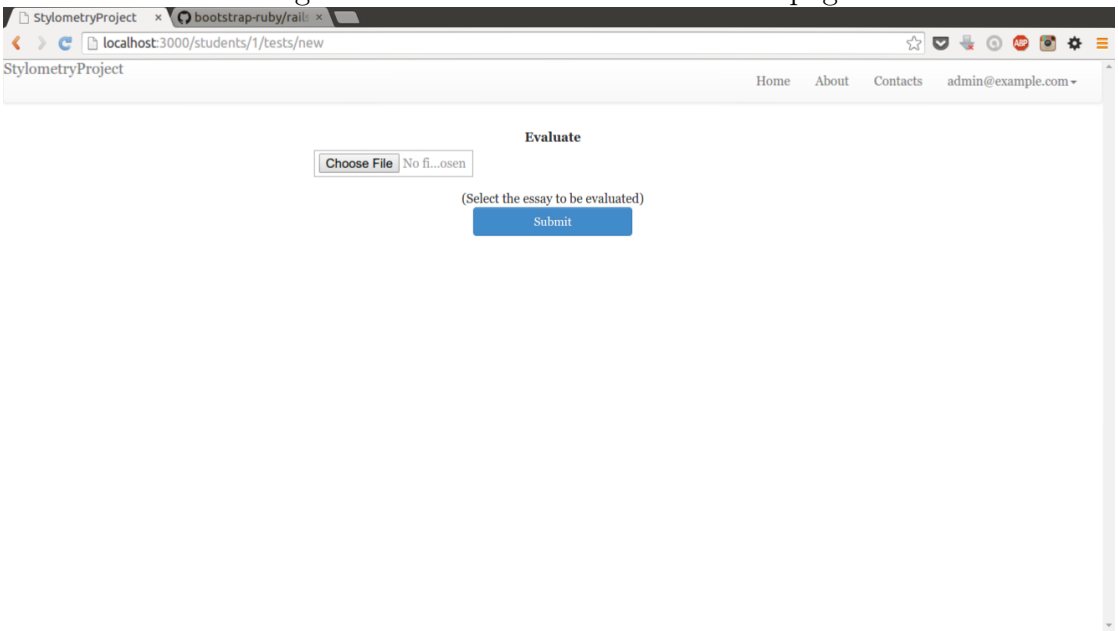
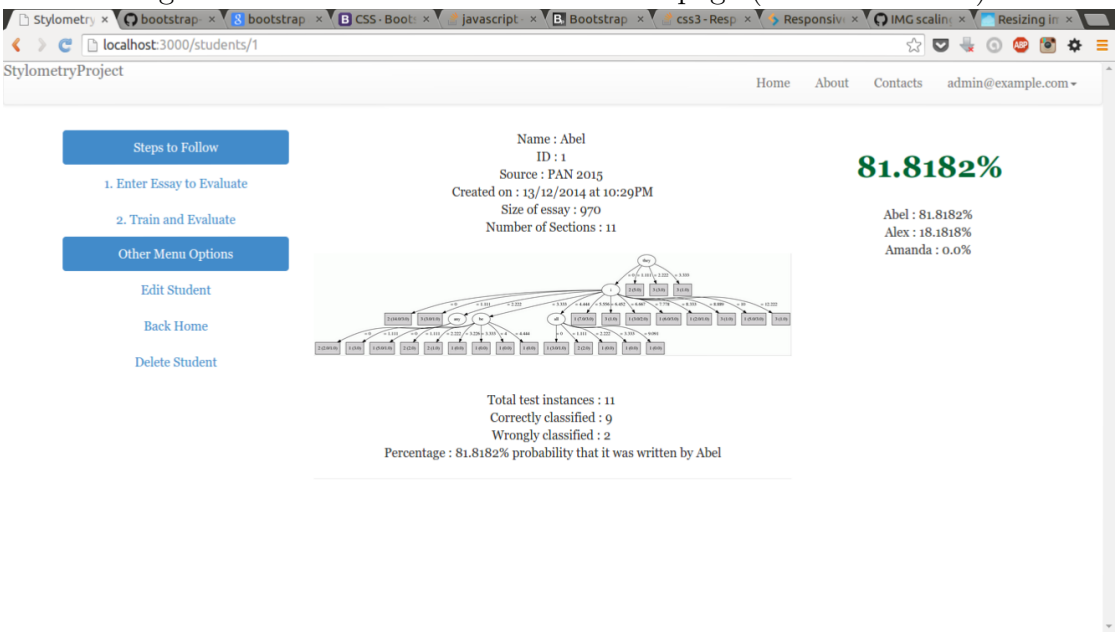


Figure 12: Screenshot of the evaluation page (after evaluation)



6 SYSTEM TESTING

6.1 Overview

This chapter deals with the testing of the system developed in the previous stage. The idea behind this chapter is to prove that the built system does what it was built to do, and with a measure of accuracy. The chapter begins with an introduction, and then an example of a test done on the developed system follows next.

6.2 Introduction

The datasets, both the training datasets and the test datasets, used in the development and testing of this system were retrieved from PAN 2015. PAN 2015 is the 13th evaluation lab on uncovering plagiarism, authorship, and social software misuse. It is the initiative of The Webis Group. The Webis Group meets challenges of the information society by conducting basic research, developing technology, and implementing and evaluating prototypes for future information systems.

The training and testing corpus provided by PAN 2015 comprises of a set of author verification problems in several languages and genres. Each problem consists of some (up to five) known documents by a single person and exactly one questioned document. All documents within a single problem instance are in the same language and best efforts are applied to assure that within-problem documents are matched for genre, register, theme, and date of writing. The document lengths vary from a few hundred to a few thousand words.

The link to the testing and training corpus is `http://www.uni-weimar.de/medien/webis/research/events/pan-14/pan14-web/author-identification.html`.

The documents of each problem are located in a separate folder, the name of which (problem ID) encodes the language/genre of the documents. The following list shows the available languages/genres, their codes, and examples of problem IDs:

Language	Genre	Code	Problem IDs
Dutch	essays	DE	DE001, DE002, DE003, etc.
Dutch	reviews	DR	DR001, DR002, DR003, etc.
English	essays	EE	EE001, EE002, EE003, etc.
English	novels	EN	EN001, EN002, EN003, etc.
Greek	articles	GR	GR001, GR002, GR003, etc.
Spanish	articles	SP	SP001, SP002, SP003, etc.

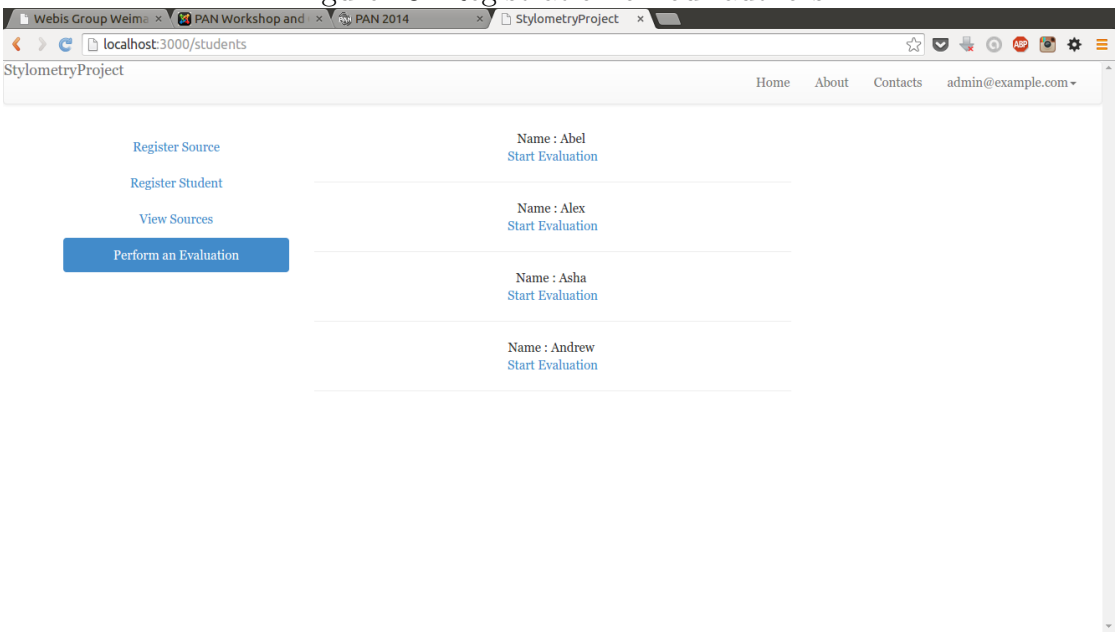
6.3 Example of a Test Performed

In this section, four problems will be chosen to test the functionlity of the developed system. EE590, EE572, EE571 and EE545 were selected randomly. For simplicity, the authors of the known.txt files in the three problems will be referred to as Abel, Alex, Asha and Andrew, respectively.

The known1.txt of the four authors will be selected as the essays used to train the model of the developed system to learn the individual linguistic styles if the authors. The subsequent known.txt files of the authors will be used to test the performance of the system and its accuracy.

The first step will be to register the authors into the system with their initial essay. Figure 13 shows the homepage of the web applicatin after the four authors have been registered.

Figure 13: Registration of four authors

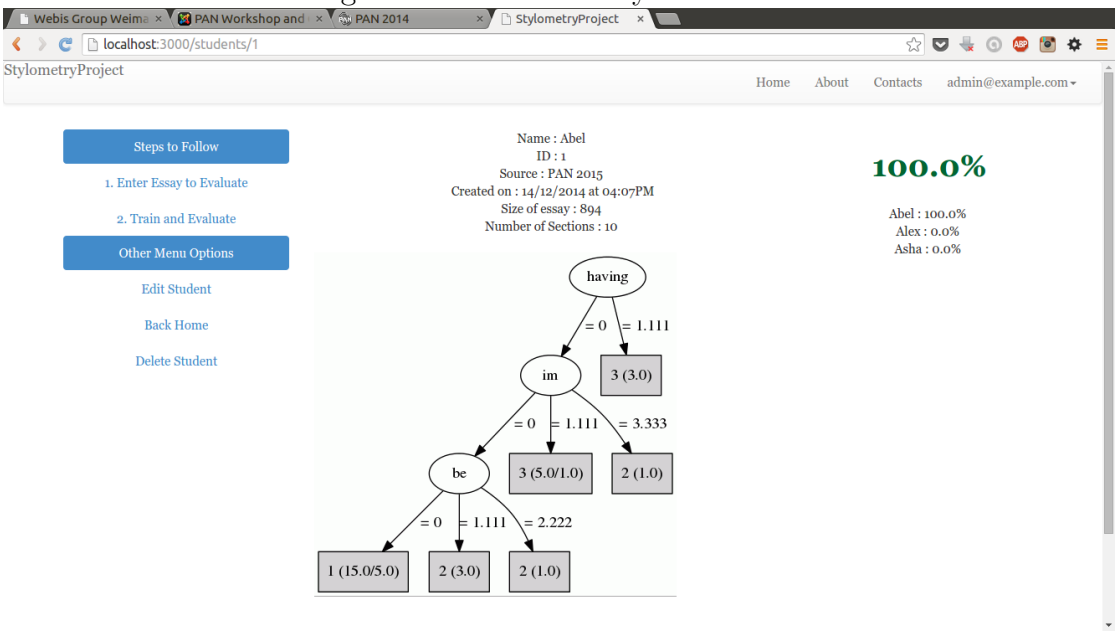


The second step is where the actual testing begins.

6.3.1 Testing of Abel (EE590)

Abel's known3.txt essay was chosen as the first file to be tested. It was input and the upon training and evaluation, the results are as shown in figure 14.

Figure 14: Abels's Essay Evaluation



6.3.2 Testing of Alex (EE572)

Alex's known4.txt essay was chosen as the file to be tested. It was input and the upon training and evaluation, the results are as shown in figure 15.

6.3.3 Testing of Asha (EE571)

Asha's known4.txt essay was chosen as the file to be tested. It was input and the upon training and evaluation, the results are as shown in figure 16.

When Asha's known2.txt essay was chosen for evaluation the results are as shown in figure 17.

6.3.4 Testing of Andrew (EE545)

Figure 18, figure 19 and figure 20 show the results of the evaluation of known2.txt, known3.txt and known4.txt of the essays belonging to Andrew.

Figure 15: Alex's Essay Evaluation

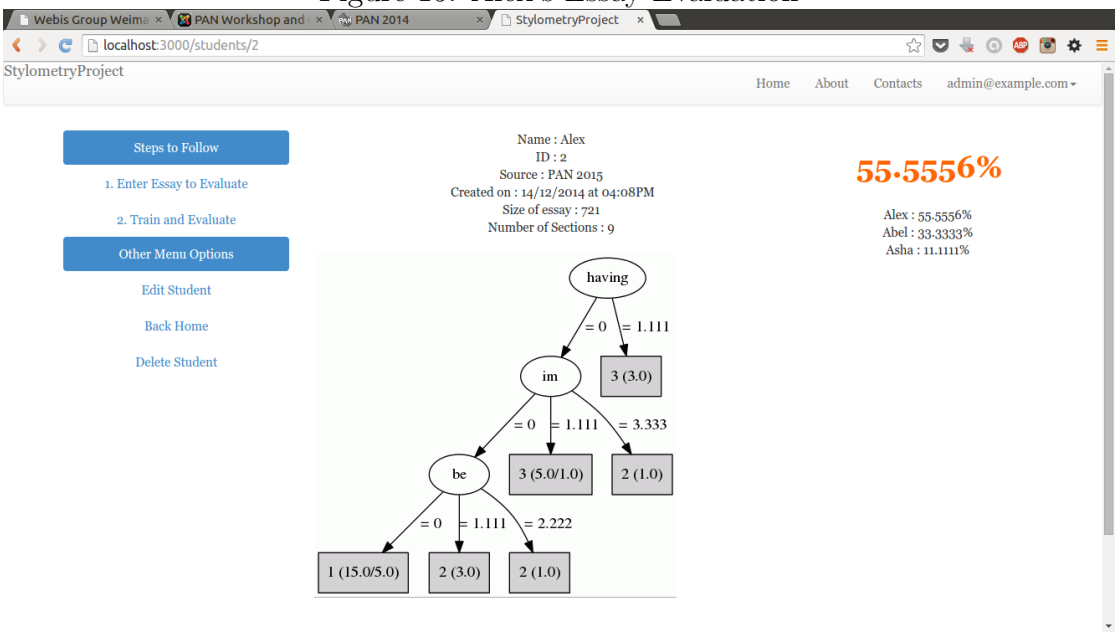


Figure 16: Asha's First Essay Evaluation

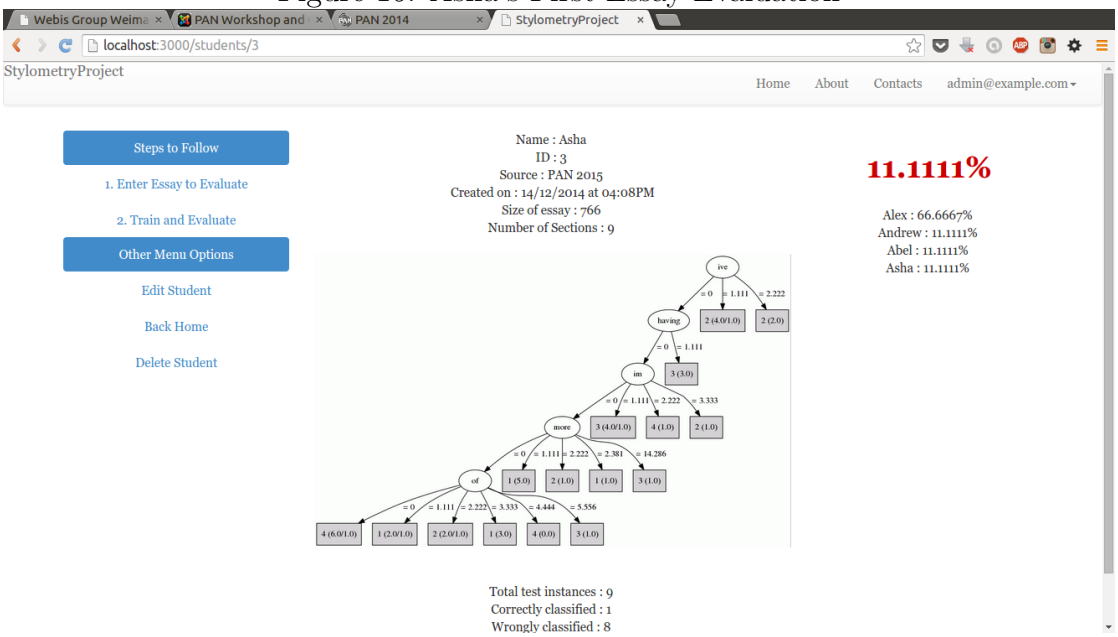


Figure 17: Asha's Second Essay Evaluation

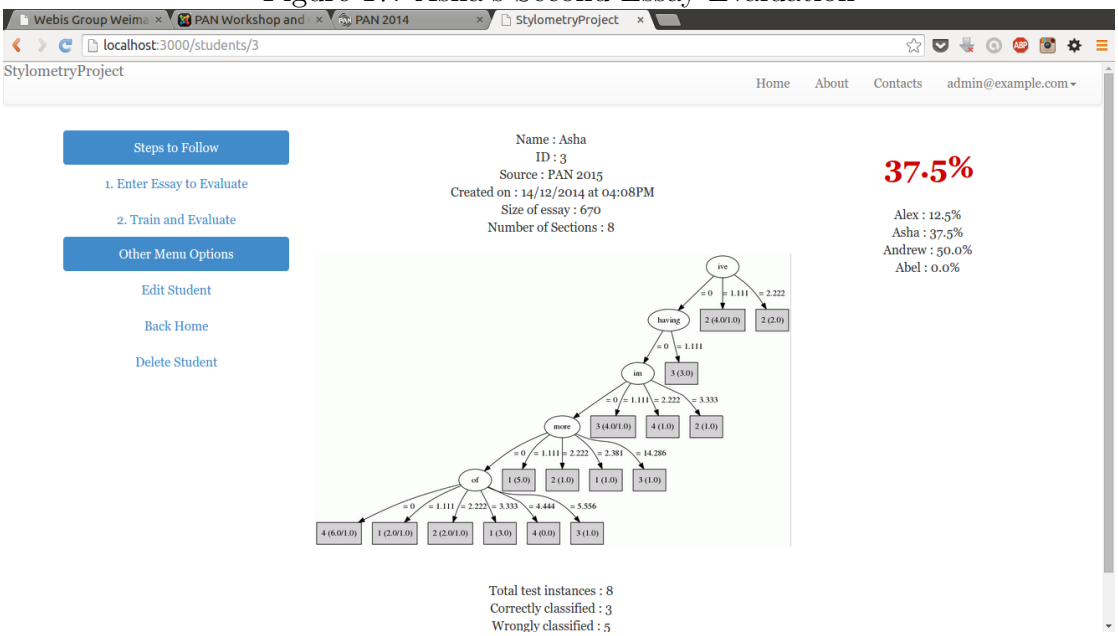


Figure 18: Andrew's First Essay Evaluation

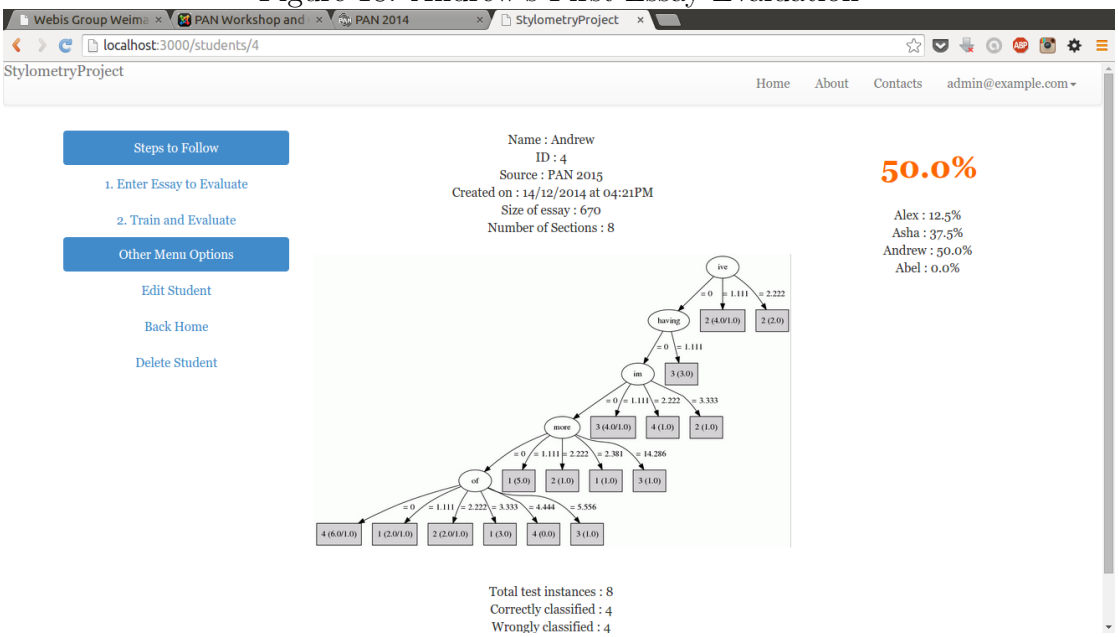


Figure 19: Andrew's Second Essay Evaluation

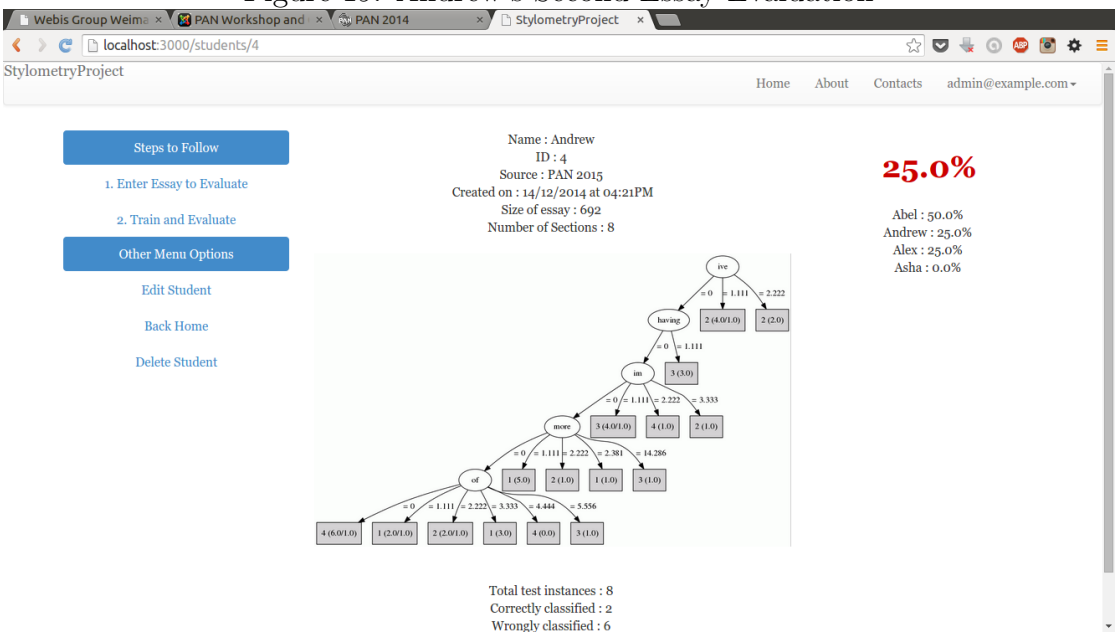
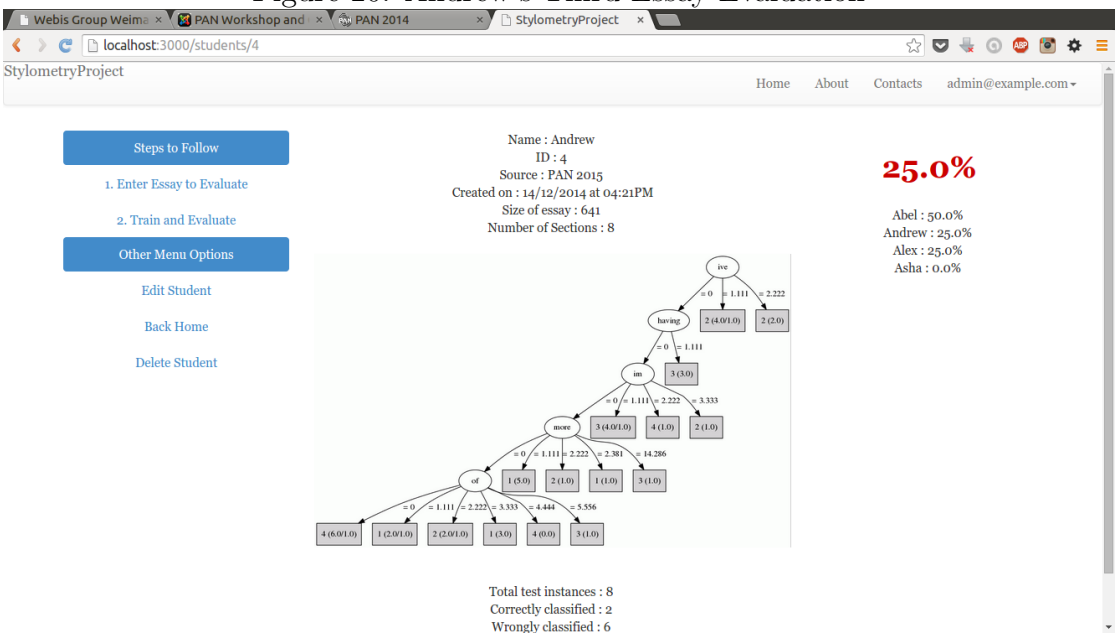


Figure 20: Andrew's Third Essay Evaluation



7 RESULTS AND ANALYSIS

7.1 Introduction

The J48 package used in this experiment extends the basic ID3 algorithm of Quinlan. It infers decision trees by growing them from the root downward, greedily selecting the next best attribute for each new decision branch added to the tree. Thus, decision tree learning differs from other machine learning techniques such as neural networks, in that attributes are considered individually rather than in connection with one another. The feature with the greatest information gain is given priority in classification. Therefore, decision trees should work very well if there are some salient features that distinguish one author from the others.

7.2 Analysis

To evaluate the classification model 10 fold cross validation and percentage split methods have been used. In 10 fold cross validation all the data has been divided into 10 disjoint set of approximately equal size. This is an iterative process. Each time 9 disjoints sets acts as a training data and one set is used as a testing data. In percentage split method 66% of the entire data has been used as training data and remaining data as testing data.

Figure 21 shows the results of the analysis.

From the diagram, classes a, b, c and represent authors Abel, Alex, Asha and Andrew, respectively

Since the attributes tested are continuous, all the decision trees are constructed using the fuzzy threshold parameter, so that the knife-edge behavior for decision trees is softened by constructing an interval close to the threshold.

Analysis does show that there does exist a unique linguistic style in each individual.

7.3 Conclusion

As much as it can be seen that there does exist an inherent and subconscious unique linguistic style per every author, the accuracy of the built model is still lacking. This can be attributed to the diversity, or lack of it, in the datasets used and/or the stylistic features used to extract data from the datasets.

Figure 21: Results and Analysis

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances          22               61.1111 %
Incorrectly Classified Instances        14               38.8889 %
Kappa statistic                          0.4783
Mean absolute error                      0.2103
Root mean squared error                  0.4367
Relative absolute error                 56.2626 %
Root relative squared error            100.8169 %
Coverage of cases (0.95 level)          63.8889 %
Mean rel. region size (0.95 level)      34.7222 %
Total Number of Instances               36

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.556    0.148    0.556      0.556    0.556      0.407    0.718     0.475     1
                 0.727    0.040    0.889      0.727    0.800      0.731    0.838     0.730     2
                 0.667    0.222    0.500      0.667    0.571      0.408    0.671     0.361     3
                 0.429    0.103    0.500      0.429    0.462      0.345    0.586     0.297     4
Weighted Avg.    0.611    0.125    0.633      0.611    0.616      0.494    0.717     0.490

=== Confusion Matrix ===

 a b c d   <-- classified as
 5 0 4 0 | a = 1
 0 8 0 3 | b = 2
 3 0 6 0 | c = 3
 1 1 2 3 | d = 4
```

8 CONCLUSION AND RECOMMENDATIONS/FUTURE WORK

This chapter gives the challenges faced during the design and implementation of the proposed system. The researcher/designer/developer of the proposed system also gives his views and opinions on futures works in the field of stylometry. Lastly, this chapter ends with a conlcusion from the researcher.

8.1 Challenges

The challenges faced during the design and implementation of this project include:

1. Time constraint. The implementation of the system took three weeks longer than was expected.
2. The designer and developer of the system failed to include punctuation marks as one of the stylistic features.

8.2 Recommendation / Future Work

Below are the recommendations for future works to be done in the field of stylometry for authorship attribution using the ID3 algorithm in decision tree:

1. This project does not take into consideration the use of punctuation marks as stylistic features in the classification of the authors (students). Future work should aim to use them so as to broaden the variety of features used and determine its effect on the accuracy levels of the system.
2. The fact that neural networks look at different features from those considered by decision trees could be put to use. A meta-learner could be trained to use both neural network and decision trees to get near perfect classifications.
3. Different set of features may be tried to see if there exists a set of feature which makes different learning techniques give the same results. Also feature extraction should be done in some care to train these learners with the most relevant features. Also, Automatic feature selecting algorithms (like winnowing) can be used to select the initial set of features.
4. This project could be extended to any number of authors (students). Currently, this project can classify only a given number of authors (students).

The next step could be to generate abstract general inductive rules that can be used in all author identification problems instead of learning the ways to separate a group of n chosen authors (students).

5. Future works may try to incorporate attributes such as age and gender of the authors (students), and analyze their effects on the training and classification of the decision tree model, and on the evaluation of essays.

8.3 Conclusion

This project attempts to recognize different authors (students) based on their style of writing.

Most of the previous studies in the field of stylometry for authorship attribution and identification input a large number of attributes. While more features could produce additional discriminatory material, the present study proves that artificial intelligence provides stylometry with excellent classifiers that require fewer input variables than traditional statistics. Hence the conclusion that the combination of stylometry and Artificial Intelligence, Decision Trees to be more specific, will result in a useful discipline with many practical applications.

References

[1] H. Moore. *Campus cheaters hire custom essay writers to avoid detection*, CBC News. 26 February, 2014

[2] J.Rudman. *The state of authorship attribution studies: Some problems and solutions*, Computers and the humanities, pg 351-356, 1998

[3] P. Sharma and M. Kaur. *Classification in Pattern Recognition*, International Journal of Advanced Research in Computer Science and Software Engineering, vol. 3, no. 4, April 2013

[4] H. Maurer, F. K. *Plagiarism - a survey*, Journal of Universal Computer Science, vol. 12, no. 8, pg 1050-1084, 2006

[5] D. Adair. *The Authorship of the Disputed Federalist Papers*, The Wiliam and Mary Quarterly, vol. 3, pg 97-122, 1994

[6] S. Y. Sohn. *Meta-Analysis of Classification Algorithms for Pattern Recognition*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 21, no. 11, November 1999

[7] Ramyaa et al. *Using Machine Learning Techniques for Stylometry*, Artificial Intelligence Center

[8] H. Fearn. *Plagiarism software can be beaten by simple tech tricks*, Times Higher Education. 20 January, 2014

[9] J. R. Quinlan *Induction of Decision Trees*, Centre for Advanced Computing Sciences, New South Wales Institute of Technology. 1985

[10] Lakshmi et al *A Study on Author Identification through Stylometry*, International Journal of Compute Science and Communication Networks. 2012

[11] H. Hanlein *Studies in Authorship Recognition: a Corpus-based Approach*, Peter Lang. 1999

[12] K. Calix et al *Stylometry for E-Mail Author Identification and Authentification*, Proceedings of CSIS Research day, Pace University. May 2008

[13] p. Juola *JGAAP: A Modular Software Framework for Evaluation, Testing, and Cross-Fertilization of Authorship Attribution Techniques* , Duquesne University. August 2009

[14] M. Brennan et all *Practical Attacks Against Authorship Recognition Techniques*, Dept. of Computer Science, Drexel University.

[15] C. Prendergast *The Fighting Style: Reading the Unabomber's Strunk and White*, College English. 2009

9 Appendix

Contents of Stopwords.txt

a, about, above, after, again, against, all, am, an, and, any, are, aren't, as, at, be, because, been, before, being, below, between, both, but, by, can't, cannot, could, couldn't, did, didn't, do, does, doesn't, doing, don't, down, during, each, few, for, from, further, had, hadn't, has, hasn't, have, haven't, having, he, he'd, he'll, he's, her, here, here's, hers, herself, him, himself, his, how, how's, i, i'd, i'll, i'm, i've, if, in, into, is, isn't, it, it's, its, itself, let's, me, more, most, mustn't, my, myself, no, nor, not, of, off, on, once, only, or, other, ought, our, ours, ourselves, out, over, own, same, shan't, she, she'd, she'll, she's, should, shouldn't, so, some, such, than, that, that's, the, their, theirs, them, themselves, then, there, there's, these, they, they'd, they'll, they're, they've, this, those, through, to, too, under, until, up, very, was, wasn't, we, we'd, we'll, we're, we've, were, weren't, what, what's, when, when's, where, where's, which, while, who, who's, whom, why, why's, with, won't, would, wouldn't, you, you'd, you'll, you're, you've, your, yours, yourself, yourselves,

loadEnvironment()

```
def loadEnvironment
    #Load Java Jar
    dir = "./weka.jar"

    #Have Rjb load the jar file, and pass Java command line arguments
    Rjb::load(dir, jvmargs=["-Xmx1500m"])
  end
```

set_sessions_to_nil()

```
def set_sessions_to_nil
    session[:dtreeString] = nil
    session[:classname] = nil
    session[:wordCount] = nil
    session[:numberOfGroups] = nil
    session[:correctlyClassified] = nil
    session[:wronglyClassified] = nil
    session[:percentage] = nil
    session[:totalTestInstances] = nil
    session[:name] = nil
```

```
    session[:startedEvaluation] = nil
    session[:percentages] = nil
  end
```

hash_function(array)

```
def hash_function(array)
    counter = Hash.new(0)
    array.each {|i| counter[i] += 1}
    counter
  end
```