2010

Who wrote the Letter to Hebrews?

Data mining for detection of text authorship

Progress Report

13th Oct 2010

Supervisor: Derek Abbott

Co-supervisor: Brian Ng, Matthew Berryman, Maryam Ebrahimpour

Jie Dong, Leng Tan, Tien-en Phua
University of Adelaide
10/13/2010

Contents

List Of Figure

List Of Tables

Executive Summary

A group of three students has been assembled to undergo a final year project on data mining for authorship detection in a goal to uncover the true authorship of the letter to the Hebrews.. The team members that will work in collaboration in the project consist of Tien-en Joel Phua, Leng Yang Tan and Jie Dong. This document outlines the progress of the project and the work that have been done and contribute by each team members in detail. There are three main sections of this report: Progress so far, Future approach of the project and Project management.

This document encompasses the project management strategies implemented for the success of this project and provides a detailed guide through which the progress of the project can be understood. Using this document each member can evaluate the work and progress status of other team members and also have a clear understanding of the following tasks and deliverables that need to be accomplished.

By reading this document, the overall progress of the project is known and team members can evaluate and determine whether it is necessary to reschedule some task in order to achieve the desired dateline. This document provides sufficient information on the status of the project.

Project Aim

The project aims to solve the controversy "Who wrote the Letter to Hebrews?" The team intends to further enhance three extraction algorithms, Function Word Analysis, Word Recurrence Interval (WRI) and Trigram Markov, which have been shown to produce relatively satisfactory results, as compared to data compression, in terms of authorship detection and compare its effectiveness to existing algorithms. The team plans to utilize a Support Vector Machine (SVM) to develop a classification model that would be able to accurately classify a disputed text to its author using a database of undisputed texts. With this model, the team would be able to present an accurate hypothesis to the controversy "Who wrote the letter to Hebrew?" In addition, if time permits, the team would aim to verify the authorship of other controversial texts such as The Federalist Paper and the works of Shakespeare. Furthermore, the team would like to further develop our algorithm to applications such as source code plagiarism detection and future search engines.

Project Background

The author of the letter to Hebrews has been wavering for over 1,800 years. Numerous authorship techniques have been applied to the text but results have often been inconclusive or have only been able to show that it is most likely that Paul of Tarsus or Apostle Paul was not the author of Hebrews. In this project, the team aims to further enhance three existing extraction algorithms, namely Function Word Analysis (FWA), Word Recurrence Interval (WRI) and Trigram Markov, in order to identify the author of the letter. In this project, the team aims to develop a classification model using a Support Vector Machine (SVM), which has been demonstrated to be exceedingly accurate and be able to contribute significant evidence regarding the author of the letter to Hebrews.

Project Requirement and Specification

The project team aims to build a classification model using a Support Vector Machine (SVM) with either a chosen extraction algorithm or possibility a combination of algorithms that would be able to associate a disputed text to its original author.

A disputed text is an article or any piece of writing whose authorship is uncertain. So for example if we have a disputed text, Text C, that is suggested to link to either author A or author B, we would build up our classification model by entering a set of training data to SVM to build up our classification model. Our set of training data consists of several texts that have been undisputedly claimed to be written by either author A or B.

As shown in Figure 1, for example, Texts A1 to A3 written by author A and Texts B1 to B3 written by author B are used as our training data. In actual fact, our training data would vary depending on the accuracy necessary. Also shown, the testing data set, Text A4 and Text B4, written by their respectively author, is used to determine the accuracy of our classification model. And likewise, in actual fact, our testing data set would be larger than a sample size of 1. If Text A4 and Text B4 are classified correctly, into their respective authors, we can safety assume that our classification model is functioning accurately. Thereafter, we enter our disputed text, Text C, into our extraction algorithm, followed by our classification model to determine the actual author. If Text C was not able to be classified under author A or author B, we can either increase our set of training data to build up our classification model further or we can assume that Text C was not written by neither author A or author B. In addition, for the classification model to function as accurately as possible, it would be worthwhile to have training and testing sets that is as large as possible.

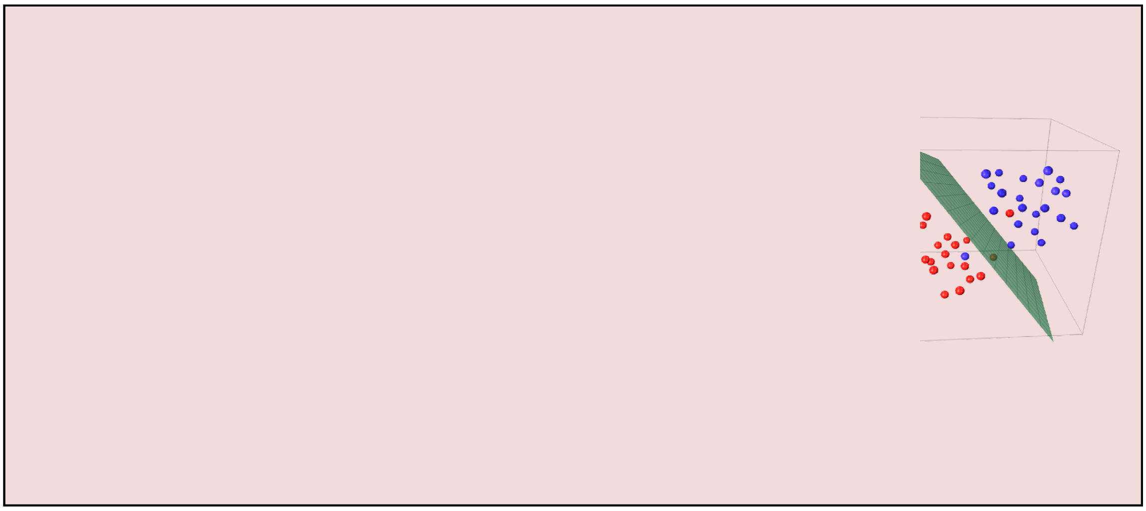

Figure 1: Overall process of the classification model of the data

Progress So Far

Research

Function Word Analysis

Typically, texts are made up from a combination of both content and function words. Function words (or grammatical words or auto semantic words) are words that have ambiguous meaning and serve to express grammatical relationships with other words within a sentence, or specify the attitude or mood of the speaker. Function words might be prepositions, pronouns, auxiliary verbs, conjunctions, grammatical articles or particles. Each function word either gives some grammatical information on other words in a sentence or clause, and cannot be isolated from other words, or it may indicate the speaker's mental model as to what is being said. [1]

The use of function words in authorship attribution is appealing as it forms the writing style of an author. In addition, their incidence is often due to authorial style and are not affected by the content of the text [4]. In contrast, content words are highly correlated with the document topics and are not suitable for authorship attribution [2]. Two authors writing on the same topic or about the same event may share many words and phrases. [4] For example the Gospel of Matthew, Gospel of Mark and the Gospel of Luke have many appearance of content words such LORD, God and salvation.

The length of a document influences the frequency of occurrence of the functional words and also their sole presence.

In 2005, a student from the University of Adelaide identified thirty function words and counted the appearance of these words in the different text by different author in the New Testaments and applied it to attribute the author of the letter to the Hebrews.

And	The	Of	To	They
That	He	In	Him	Unto
Them	A	Was	With	When
I	Paul	Which	For	All
Had	Were	God	Said	His
We	This	From	But	Not

Table 1: List of Function Words used by Talis

In addition, Mosteller and Wallace [3] identified about 70 function words and applied them in the analysis of the Federalist Papers and produced conclusive results that attributed the text to the authors.

A	Do	Is	Or	This	All	Down
It	Our	To	Also	Even	Its	Shall
Up	An	Every	May	Should	Upon	And
For	More	So	Was	Any	From	Must
Some	Were	Are	Had	My	Such	What
As	Has	No	Than	When	At	Have
Not	That	Which	Be	Her	Now	The
Who	Been	His	Of	Their	Will	But
If	On	Then	With	By	In	One
There	Would	Can	Into	Only	Things	Your

Table 2: List of Function Words used by Mosteller and Wallace

The frequency of occurrence of the function words are calculated and analyzed using one of the classification models available, namely Naïve Bayesian, Bayesian networks, nearest-neighbor method, k-nearest neighbor, decision trees, principal component analysis, linear discriminate analysis and support vector machine. The support vector machine has proven to be the most accurate classification model and thus would be used in this project to attribute the text to the author of the letter to the Hebrews.

Word Recurrence Interval

A more statistical approach should be used for authorship detection; hence the data extraction algorithm Word Recurrence Interval (WRI) was chosen. For this algorithm, the WRI is defined as the number of words in between successive occurrences of a keyword. Furthermore, a set of keywords in the text is selected based on the number of times the keyword appears in the text. Thereafter a set of scaled standard deviation are obtained from the chosen keywords.

In the context of authorship attribution, this algorithm was chosen as it eradicates the dependency of word frequency which characterizes the word distributions, thus utilizing a more statistical approach to the analysis of the text. The length of the text is an important contributor to the result, hence the length for all text will need to be kept constant.

Upon researching on a similar project that was conducted by Talis (a past year student of the University of Adelaide in 2005) it was concluded that using WRI for data extraction and plotting graphs of scaled standard deviation of WRI vs. log 10 (rank) does not give satisfactory results. Instead another type of data classification should be incorporated. For this reason, the Support Vector Machine (SVM) was utilized for this project and showed relatively good results in past year research.

Trigram Markov

Markov chains are widely used in a variety of areas in mathematics and engineering. Previous study shows that it is a useful tool in stochastic text generation. Specifically, Markov n-gram models are very powerful in statistical natural language processing, and have been shown through abundant experiments to be extremely effective in creating language models which are a core component in modern statistical language application.

Technically, a Markov Chain is defined by a set of states and transitions. It has the memory less property which means occurrence of future states does not depend upon past states, but only on the current one. Trigram Markov Chain is a particular example in this class. It indicates that the occurrence of the coming state only depends on its previous two states.

In the context of authorship attribution, Trigram Markov Chain assumes that the probability of the next letter or word (or character in some other languages) is related only to the two letters or words before it. In mathematics, it is represented as:

$$P(X_n|X_{n-1}, X_{n-2}, \dots, X_1) = P(X_n|X_{n-1}, X_{n-2})$$

With the above equation, a vector which contains the states and transitions information calculated based on each text document is formed. It is believed that texts written by the same author may have similar vectors. Therefore, these characteristic vectors can then be used for classification.

In 2005, Talis worked on the trigram model for authorship detection. He calculated all states and state transitions probabilities in each text and determined text authorship through the entropy method used before by Khmelev. His results showed a clear upward trend in the classification accuracy as the size of the training data was increased. He also suggested that a classification accuracy of 88.3% was achieved using 12 texts per author as training data however classification accuracy obtained with a greater size of training data was not explored.

In our project, the effectiveness of support vector machine which is a widely used classification technique will be explored and compared with Talis's approach.

Extraction Algorithm Programming

Function Word Analysis

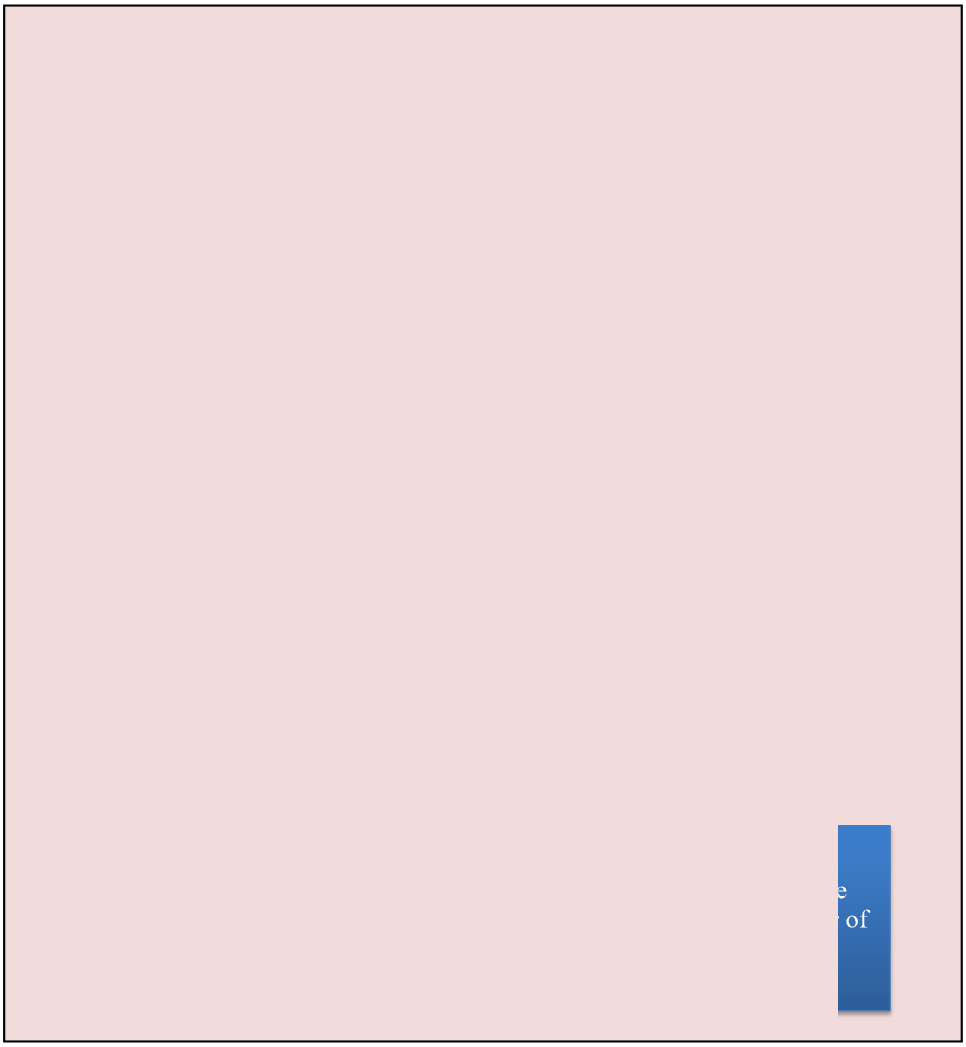

Figure 2: Block Diagram of Function Word Analysis

Step 1

The program will prompt the user if he/she would like to enter in a choice of function words to be used in the analysis. If the user prefers not to use this function, he/she can kindly decline by entering “N” in response to the prompt.

Step 2
setFunction()

This module will create a function word object for each function word. It creates two variables for each object, namely a name label and an occurrence counter. It also creates an array list to store the function words objects.

Step 3
removePunctuation()

The input for this module would be a string containing the file name that the user wishes to do a text edit. This module will pass through the text and identify punctuation symbols such as line feed, tabs and the following symbols enclosed in the brackets. (,./;[]\\=-0987654321`~!@#$%^&*()_+{}|:<>?\"\')
It will then remove the punctuation marks and create a new file containing only text with the punctuations removed.
The output of this module is a file with the file name named as the original input string with the word “Modified” concatenated in front.

Step 4
analyzeFile()

This module will analyze the entire text and count the number of occurrences of each function word that is contained in the array list.

Step 5
write Output()

This module will create a ASCII file displaying the number of occurrence of the respective function word. An example of the file is shown in the figure below.

Step 6
writeOutputSVM()

This module will create a text suitable for SVM input.

```
'and' appears : 1751
'the' appears : 1557
'of' appears : 794
'to' appears : 575
'they' appears : 389
'that' appears : 389
'he' appears : 381
'in' appears : 334
'him' appears : 316
'unto' appears : 308
'them' appears : 302
'a' appears : 261
'was' appears : 247
'with' appears : 242
'when' appears : 224
'i' appears : 219
'paul' appears : 132
'which' appears : 208
'for' appears : 199
'all' appears : 192
'had' appears : 175
'were' appears : 178
'god' appears : 175
'said' appears : 158
'his' appears : 146
'we' appears : 134
'this' appears : 147
'from' appears : 141
'but' appears : 144
'not' appears : 133
Total number of words text contains : 24282
```

Figure 3: Output of Function Word Analysis

Word Recurrence Interval

The data extraction algorithm Word Recurrence Interval(WRI) was implemented using Java as it is the most suitable programming language for the team members. The main purpose of the algorithm was to accept a text file and produce an output text file which shows the results such as keywords and standard deviation. A flow chart (below) for the algorithm was made before implementation at the initial planning stage for simplicity purpose.

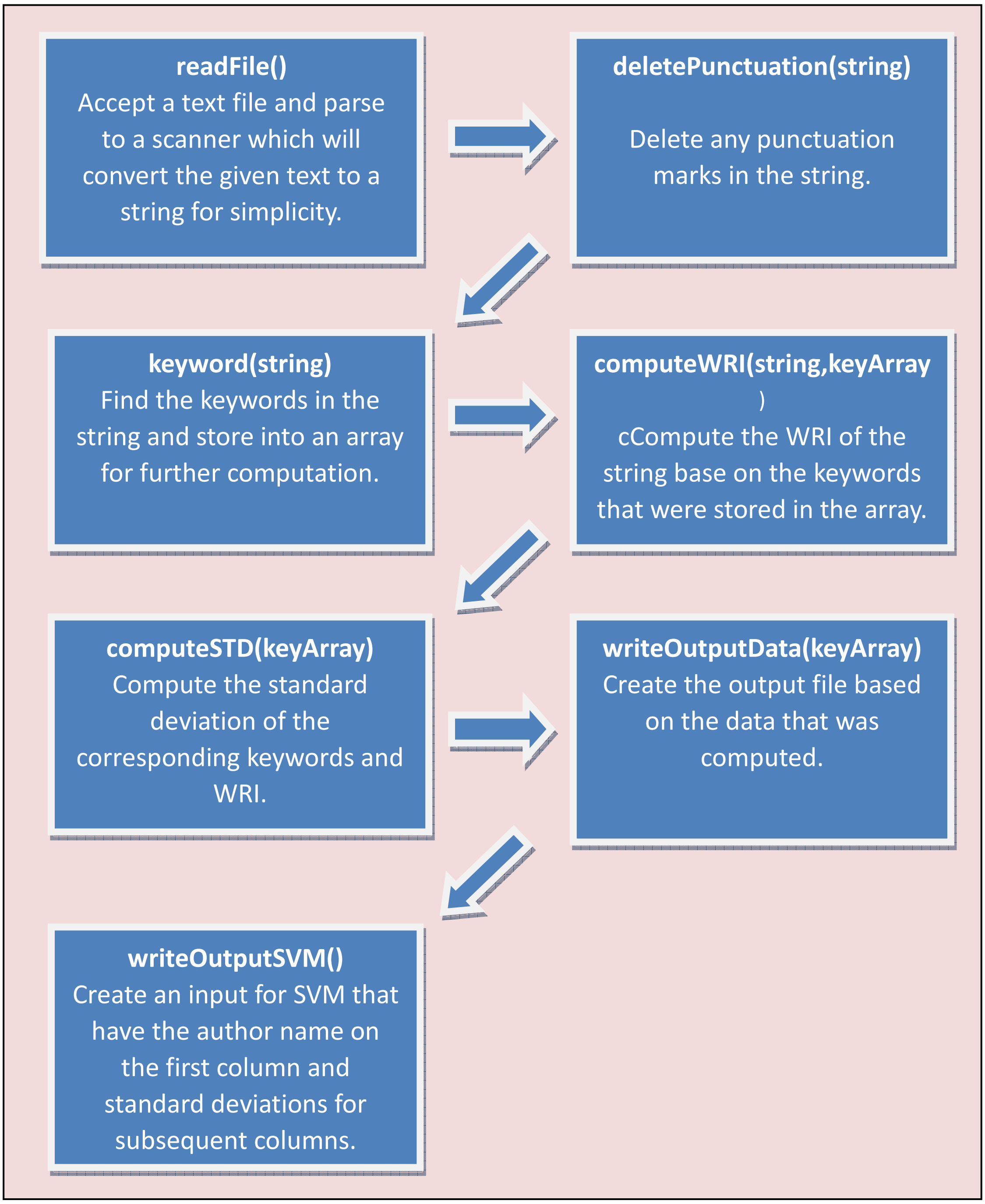

Figure 4: Algorithm of Word Recurrence Interval

This design and approach was chosen for numerous reasons and the function of the algorithm is explained below.

Step 1
readFIle()
The conversion of the text to a string would simplify the manipulation in the text prior to data computation.

Step 2
deletePunctuation(string)

It was decided by the team that the punctuation marks in the text would not provide any benefits in authorship attributions and hence the method was used to remove any punctuation marks including numbers in the text.

Step 3
keyword(string)

An array that stores all the keywords and relevant variables corresponding to the keywords in the text were used for ease in data storage. This method initializes the variables needed for computation of WRI and standard deviations.

Step 4
computeWRI(string, keyArray)

This method is used to recursively extract a keyword from the array and compute the WRI of the keyword based on the text. The result is saved back into the array. The variable would then be used to compute the standard deviations.

Step 5
computeSTD(keyArray)

This method is similar to computeWRI() except that this method computes the standard deviation. Initially the method for computing the WRI and standard deviation were combined together, however it was separated for structural design and simplicity.

Step 6
writeOutputData(keyArray)

This method creates data information of a specific text showing the keywords and corresponding WRI and standard deviations. This result would be used as part of the input for SVM.

Step 7
writeOutputSVM()

This method creates an input file to SVM based on the results obtained from each text to create the training and test data.

A short example of the output file is shown below:

Keyword	WRI	Number of times	Standard Deviation
are	4404	20	212.1439
My	4824	21	250.9619
So	4381	21	179.6636
Would	4474	22	260.2932
Holmes	4758	26	271.5872
Said	4851	27	196.6155
has	3968	28	228.0213

Table 3: Output file showing relevant data of a given text

It is necessary to convert and compile all of the processed data to another output as the input for the SVM. Hence the method "writeOutputSVM" was created to accomplish this task. A short example is shown below:

Number of texts: 52
Number of disputed texts: 1
Data dimensions: 5

Author	X1	X2	X3	X4	X5
AD	212.1439	250.9619	179.6636	260.2932	271.5872
AD	226.8029	253.9152	113.4909	185.0432	189.5957
BB	170.4921	179.5047	189.5695	166.0652	310.1569
BB	236.9008	222.5544	211.0101	320.4586	145.0703

Table 4: Output file from the algorithm as the input for SVM

Trigram Markov

The Trigram Markov extraction algorithm was implemented using JAVA. Up to the current stage, three prototypes of extraction were implemented. They are shown in the following table:

Prototype version	Prototype name	Algorithm Description	Implement environment
v0.1	Simple trigram model	Only consider trigram effect which is the occurrence of current word that only depends on two previous words	Java file
v0.2	Hidden Markov train model	Take into account the effect of bigram and unigram	Java file, makefile
v1.0	Modified Hidden Markov train model	Do not use all words that appearin a text as inputs, instead choose a specified number of function words as inputs	Eclipse

Table 5: A table showing the different prototype of the algorithm

The first prototype of this algorithm make the assumption that only the previous two words have any effect on the probabilities for the next word. The probabilities are calculated using the following formula:

$$P_e(w_i|w_{i-2,i-1}) = \frac{C(w_{i-2,i})}{C(w_{i-2,i-1})}$$

where $C(w_{i-2,i})$ represents the number of times that this trigram appeared in the text and $C(w_{i-2,i})$ represents how many times the bigram containing previous two words appeared. Hence their ratio indicates the probability of the third word that appears after previous two. However, while this model is applied, it shows that there are almost no common trigrams among texts. Past research shows this problem on sparse data as well. In other words, suppose that the relevant statistics were collected for our trigram model and then apply it to a new text in which a trigram occurs that never appeared in the training corpus. Then the third word coming after the first two would have a zero probability, resulting in very poor cross entropy.

In this case, a second prototype was developedas a solution to this problem to smooth the probabilities by using both the bigram and unigram probabilities. It is defined using the following formula:

$$P(w_i|w_{i-2,i-1}) = \mu_1 P_e(w_i) + \mu_2 P_e(w_i|w_{i-1}) + \mu_3 P_e(w_i|w_{i-2,i-1})$$

where P_e is the unigram, bigram and trigram probabilities calculated in a similar way as defined in the above equation . The three non-negative coefficients µ1, µ2, µ3 have a sum of 1. If we assume that most of the time we do have trigrams and that they yield a more accurate assessment of the probabilities than bigrams or unigrams, then µ3 should be much higher than the other two such that it dominates the probability calculation. Their values can be determined by a scheme called hidden Markov models (HMM). Unfortunately, this algorithm has not been implemented to automatically determine their value according to the input texts. Instead, values of 0.1, 0.3, and 0.6 were assigned to them for convenience and testing purpose at current stage. While running the program with an input of 30 texts, the amount of common trigrams among texts wasstill not satisfactory. Even though tolerances of missing trigrams are allowed in a certain number of texts (For example, for 30 input texts, common trigrams are defined to be those that appear in at least 26 texts, so the tolerance is 4), the dimension of output vector for each text is still small. Hence statistics that can be used for training the support vector machine is insufficient resulting in an inaccurate prediction on disputed texts.

Looking at Talis's algorithm, it shows that Talis removed non-functional words first then looked for common trigrams among texts. As mentioned in "Function words analysis" section, function words (or grammatical words or auto semantic words) are words that have little lexical meaning or have ambiguous meaning, but instead serve to express grammatical relationships with other words within a sentence. Hence, texts with only function words efficiently represent an author's writing style and could be very helpful for classification. As a result, prototype three was inherited from second model and added a function before looking for trigrams and calculating their probabilities. The aim of this additional function is to compare input texts and collect a list of most frequently occurring words (function words) in these texts, followed by the formation ofa new text by removing other words that did not appear in the list from the original text. In this way, the size of common trigrams were increased to provide more information to SVM for more accurate classification.

This prototype is implemented using Eclipse which is a convenient software for this group project. Three extraction algorithms (function word analysis, WRI and trigram Markov model) will be combined into one project folder later. Hence using Eclipse the code structure can be standardized and the program can be integrated more easily.

The flow chart below shows the basic structure of this algorithm:

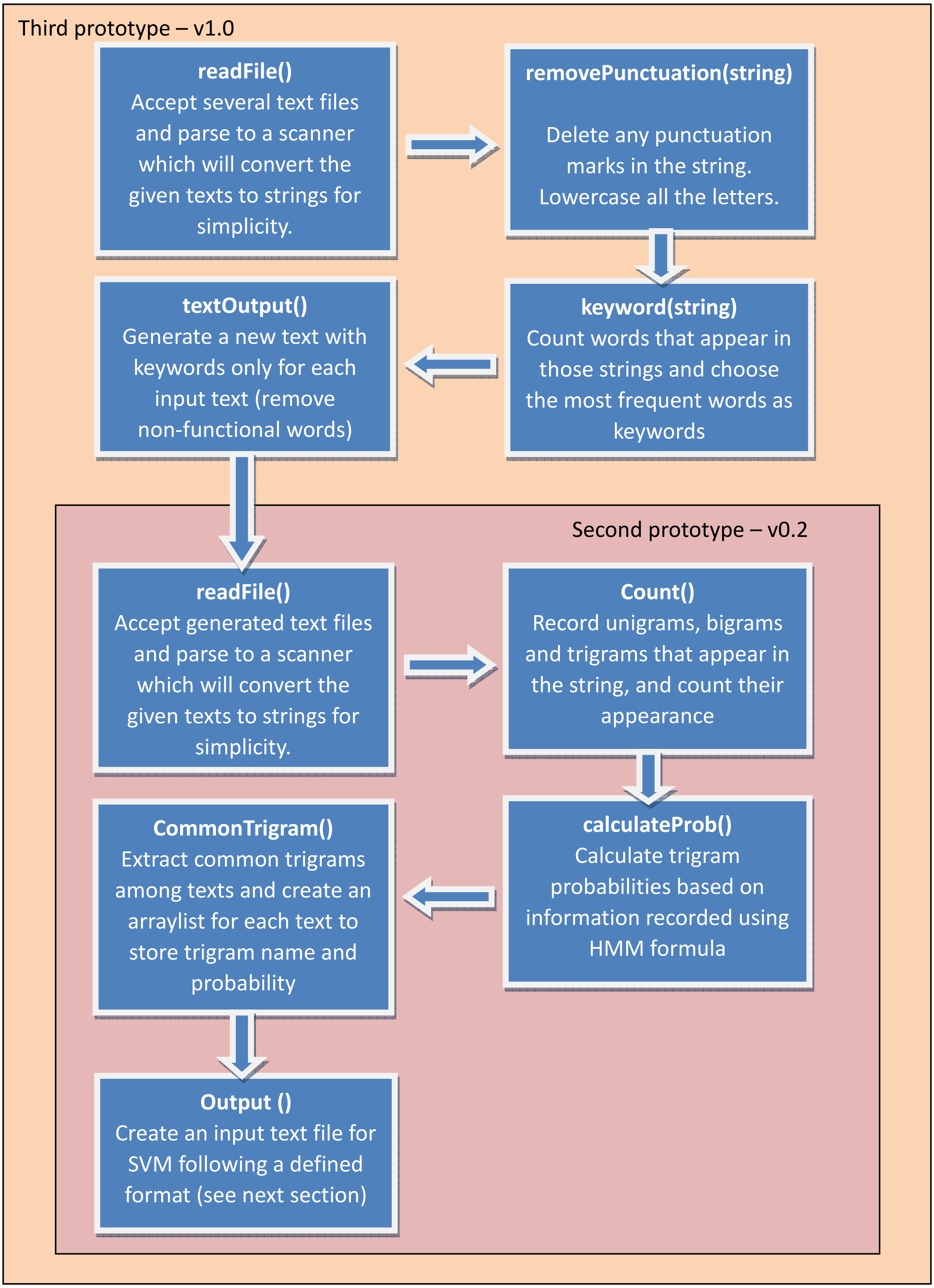

Figure 5: A flowchart of the Trigram Markov algorithm

Here is an example using an algorithm described above with 52 input texts (including 4 disputed texts randomly chosen) from Davis R.H and Grey Z. The number of functional words is set to 30. By running the program, 30 words with most frequent occurrences were picked and listed in descending order as follow:

The	and	Of	To	A	He
In	Was	I	That	His	It
Had	You	With	Her	She	For
As	Him	At	Not	On	But
From	Me	They	One	Be	Were

Table 6: 30 most frequent occurrences words

Based on these key words, the remaining texts in the file are deleted leaving only the key words. After comparing all texts, 8 common trigrams were found with 0 tolerances:

a in the	and the of
the of a	the to the
the and the	of the and
in the of	the of the

Table 7: Common trigrams with 0 tolerances

The probability of each trigram was then calculated. Finally, a 52 by 8 probability table was produced and wrote into a txt file with a format described in next section for SVM classification.

SVM Implementation

As introduced in last section, JAVA program implementing three extraction algorithms produced an output text file which will be used as Matlab SVM input. The input file follows a defined format as follows:

%Header field
Number of texts:
Number of disputed texts:
Data dimension:

%Blank line

%Data field

Author	prob	prob	prob	prob	prob	prob	prob	prob …
Author	prob	prob	prob	prob	prob	prob	prob	prob …
Author	prob	prob	prob	prob	prob	prob	prob	prob …
Author	prob	prob	prob	prob	prob	prob	prob	prob …
Author	prob	prob	prob	prob	prob	prob	prob	prob …

Figure 6: Format of the input file for SVM

Note:

- Number of texts includes both training texts and disputed texts. It also indicates the number of rows contained in the data field.
- Data dimension represents the number of probabilities / columns used
- In the data field, numbers and strings are separated by a single tab.
- Data for disputed texts are always listed in the bottom rows of the data field.

Support vector machine (SVM) was supported since Matlab 2008 version. It provides two functions for training and classifying – svmtrain and svmclassify. With these two functions and SVM input file, the SVM program performed the classification job as shown in the following flow chart:

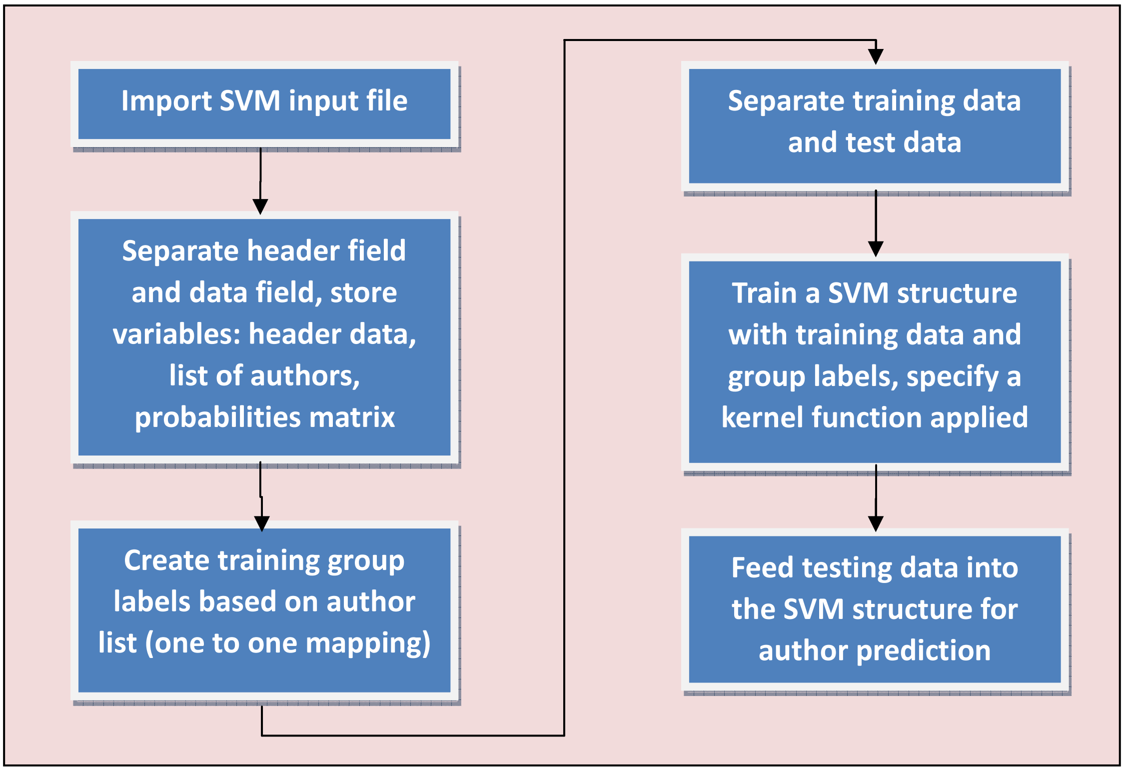

Figure 7: Process of data classification in SVM

Testing

After the completion of the data extraction algorithm, the algorithm was tested by inputting a very short text and manual calculations. This ensures that each algorithm is working as expected and also verifies if there are any mistakes.

Technical Challenges

During the implementation of the algorithm, several technical issues arose and challenges had to be overcome. In this section, the obstacles that were faced and the solutions that used to solve the problems are stated.

Technical Issue 01

The challenge in using function word analysis is in determining the choice of function words to use for the analysis of text. It is necessary to conduct a first pass to identify all the words in the text and filter out all the content words. Function words that occur frequently are chosen. Choosing a large range of function words would consume a longer processing time and might produce inconclusive results. Therefore an analysis is required on all the text to determine the best choice of function words.

Technical Issue 02

The data storage for keywords, word recurrence interval, and standard deviation proved to be a challenge. It was discussed and resolved by introducing a new object variable for each keyword called "WRI" where the object created had the 4 variables listed below:

- String name - Store the keyword
- integer counter - Count the number of the specific keyword in the text
- an integer Arraylist - Stores the WRI in between successive keywords
- a double value - Standard deviation of the keyword

```
String name;
int counter;
Arraylist<Integer>
WRI;
double std;
```

Figure 8: Example of the WRI object constructor

Based on this design, several objects were created based on the number of keywords for a text file. An additional Arraylist of type WRI was introduced to store this objects to resolve this challenge.

Technical Issue 03

Another challenge that arose was designing a consistent input to SVM for all 3 algorithms. It was required to design an output from the data extraction algorithm that resulted in a way that it is easily fed into SVM for the purpose of data classifications. It was discussed by team members that the input to the SVM should be a text file which would have the first 3 rows stating the number of training data, the number of disputed text and the data dimensions. The probability and/or number of occurrences are listed thereafter.

Future Approach of the Project

Algorithms Integration

At current stage, the three algorithms are developed separately. In order to compare algorithms' efficiencies, the algorithms will be combined together in the same project folder using Eclipse. The designed structure for the integrated program will look like the following figure:

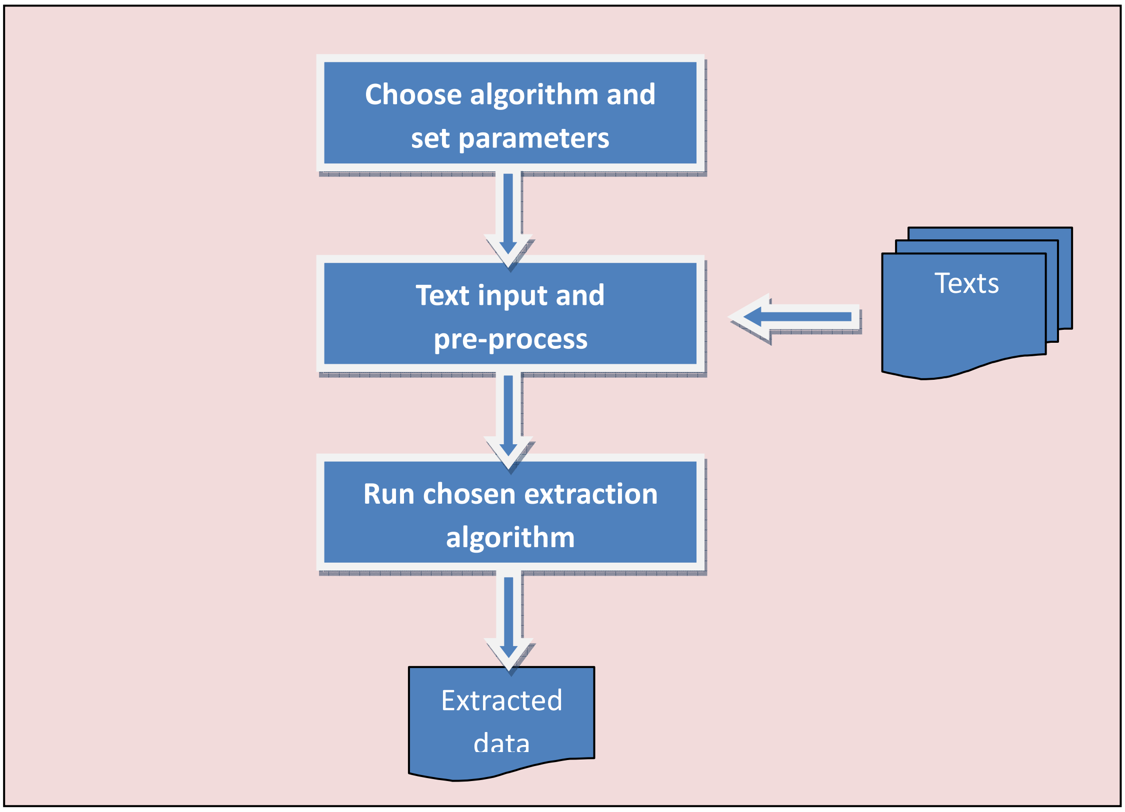

Figure 9: Design structure of the integrated program

The program is divided into three classes:

- Main Driver class - A variable field will be set to the algorithm that would be applied for data extraction. Its corresponding parameter such as number of function words, etc will also be set in this class.
- Sub-Driver class - After all user inputs are ready, the main driver class will pass control to its sub driver whose job is to import all texts needed for training and classifying. It then preprocesses all the texts for further extraction. The preprocess job includes converting each text to a single string, removing punctuations and converting all letters to lowercase, etc.
- At final stage, the program will perform specified extraction algorithm to process all texts. It will extract useful statistics and export them into a txt file which will be used for SVM classification.

Graphic User Interface

The team has decided to implement a Graphic User Interface (GUI) that combines all 3 data extraction algorithm into a single program.This would standardize the input and output of the data extraction results. The GUI will have an option called “open” where a user will be asked to input a text file or a folder for the training data. Furthermore, the GUI will also have a dropdown list to choose which data extraction should be used. In addition, The GUI would also have a display panel showing the extracted data information of a specific text file based on the selected algorithm.

Algorithm Comparison

The next step of our project is to compare the accuracy of the three algorithms’ performance in different situations for authorship detection. Factors which should be considered are size of train texts, number of key words used, text length and SVM kernel function applied. For the first test, 156 English fictional text corpuses that had been used by Talis will be adopted. 15 texts from each author will be used as training set, while the remaining ones are treated as disputed texts. The classifications results are then compared to their real authors hence calculate accuracy by equation:

$$Accuracy = \frac{Number\ of\ texts\ with\ correct\ classification}{Total\ number\ of\ "disputed"\ texts\ used\ for\ classification}$$

With different parameter combinations, statistics will be recorded into the following table:

Total Number of Input Data	Type of Algorithm used	SVM Kernel function	Number of function / Keywords used	Number of Disputed Text	Number of Text correctly classified	Accuracy

Table 8: Statistic record of each algorithm

Project Management

Milestones and Timeline

At the start of this project, the team identified key milestone as shown in Table 9. Several milestones have been met thus far, such as proposal seminar, stage 1 design document and peer review of stage 1 design document. In addition to these key deliverables to the school, additional internal milestones have been added by the team members.

Events	**Date**	**Action By**
Proposal Seminar	11th August 2010	Jie Dong, Leng Tan Tien-en Phua
Stage 1 Design Document	23rd August 2010	Jie Dong, Leng Tan Tien-en Phua
Peer Review of Stage 1 Design	30th August 2010	Jie Dong, Leng Tan Tien-en Phua
Progress Report	22nd Oct 2010	Jie Dong, Leng Tan Tien-en Phua
Interim Performance	29th Oct 2010	Jie Dong, Leng Tan Tien-en Phua
Final Seminar	2nd May 2011	Jie Dong, Leng Tan Tien-en Phua
Final Performance	23rd May 2011	Jie Dong, Leng Tan Tien-en Phua
Project Exhibition	3rd June 2011	Jie Dong, Leng Tan Tien-en Phua

Table 9: List of deliverables

Job Delegation

The workload for this project is divided to the team member base on the Work Breakdown Structure as shown in appendix C.

Referring to the work breakdown structure, each team member is responsible for one of the data extraction techniques, namely function word analysis, word recurrence interval and trigram markov. This approach provided the team with the flexibility to undergo three tasks in parallel, maximizing time and work efficiency. Furthermore, a Gantt chart (refer to appendix B) was produced to include the additional internal milestones that was decided by the team to monitor the project progress.

In addition to the work breakdown structure, the writing up of the various reports, namely the Stage 1 Critical Design Document and Progress Report was divided equally to different team members to handle a particular section. Furthermore, after the completion of the individual write up, each team member was given a follow-up task such as, compilation of the different sections by Leng Yang Tan, proof reading and editing by Tien-en Joel Phua and upload of document onto Wiki by Leng Yang Tan and Jie Dong.

Stage 1 Critical Design Document	
Abstract	ALL
Project aim	ALL
Background to the problem of the authorship of the Letter to the Hebrews	Tien-en Phua
Background and Significance	Jie Dong
Literature review	Leng Tan
Project requirements	Tien-en Phua
Three Data Extraction Algorithms - Function Word Analysis	Tien-en Phua
Three Data Extraction Algorithms - Word Recurrence Interval (WRI)	Leng Tan
Three Data Extraction Algorithms - Trigram Markov Chain	Jie Dong
Proposed Approach	Jie Dong
Milestones and Timeline	Leng Tan
Project Budget	Tien-en Phua
Reference	ALL
Appendix A: Technical Risk Analysis	Leng Tan
Appendix B: Occupational Health and Safety	Leng Tan
Appendix C: Gannt Chart	Tien-en Phua
Appendix D:Work Breakdown Structure	Tien-en Phua

Table 10: Work Breakdown for Stage 1 Critical Design Document

Progress Report	
Executive Summary	ALL
Project aim	Leng Tan
Project Background	Leng Tan
Project Requirement and Specification	Leng Tan
Progress So Far	
Research - Function Word Analysis	Tien-en Phua
Research - Word Recurrence Interval	Leng Tan
Research - Trigram Markov	Jie Dong
Extraction Algorithm Programming - Function Word Analysis	Tien-en Phua
Extraction Algorithm Programming - Word Recurrence Interval	Leng Tan
Extraction Algorithm Programming - Trigram Markov	Jie Dong
SVM Implementation	Jie Dong
Testing	ALL
Technical Challenges - Function Word Analysis	Tien-en Phua
Technical Challenges- Word Recurrence Interval	Leng Tan
Technical Challenges - Trigram Markov	Jie Dong
Future Approach of the Project - GUI	Jie Dong
Future Approach of the Project - Toolbox	Jie Dong
Project Management - Milestone and Timeline	Tien-en Phua
Project Management - Job Delegation	Tien-en Phua
Project Management - Information Management	Tien-en Phua
Project Management - Budget and Resources	Tien-en Phua
Project Management - Risk Management	Tien-en Phua
Conclusion	ALL
Reference	ALL

Table 11: Work breakdown of Progress Report

Information Management

Each team member has the responsibility to report and update their own findings on the online wiki -[[Authorship detection: 2010 group]]. In addition, the team has been meeting up fortnightly to update our progress, analyze the current stage of the project and plan the next step of our project. Minutes of the meetings can be obtained by the online wiki - [[Minutes of Meeting 2010: Who wrote the Letter to the Hebrews?]]

Project Budget & Resources

An amount of two hundred and fifty dollars was allocated to each student for this project, resulting in a total budget of seven hundred and fifty dollars.

Total allocated budget		**$750**	**Expenses Thus Far**
Expenses			
i.	Printing of research documents	$200	-
ii.	Purchase of resources	$200	
iii.	Additional resources	$100	
Total Expected Expenses		**$500**	

Printing of research documents would consist of past research done by various institutions, for the project team to analyze and evaluate the research that has been carried out up to date.

Purchase of resources would include books that have been written in regards to the author of the letter to the Hebrews. Additional resources such as online books would be purchased to use as our training and testing data to measure the accuracy of our classification model.

Additional resources include purchase of storage devices such as compact disc, to store data, software programs, handbook and reports.

Risk Management

Operation Health and Safety risk are managed to reduce the overall cost of the project and to improve performance in both the team's morale and productivity. It is important to provide a safe working environment for team members and also raise the awareness of risk to the members. Due to nature of this project, the team area of risk is constricted indoors.

The following terminology will be used in this section

- Hazard
  - A potential source of injury or ill-health
- Risk
  - A measure which combines the probability (likelihood) and possible severity (or consequences) of a hazard causing injury, illness or property damage

Hazard	Preventive Measures	Probability Rating / 10	Impact Rating / 10	Priority Score / 100
Suffer from back and neck injury due to sitting in bad posture	Ensure that position is in a upright position and obtain a comfortable chair	10	6	60
Develop hand and leg soreness due to lack of rest	Regularly stand up and walk around to exercise the limbs	5	4	20
Inadequate sleep resulting headache or migraine	Rest when required	6	5	30
Suffer from depression and anxiety due to incapability of solving program code	Seek for help if mentally road block occurs	9	4	36
Strain on visual optics due to staring too long on the computer screen	Look away from monitor every 5 minutes after working for 30 minutes	3	10	30

Table 12: Risk Analysis

Conclusion

As a conclusion, the project is progressing very well and is ahead of schedule based on the initial Gantt chart. However, additional tasks such as GUI and SVM implementation might consume more time than it was initially assumed, thus the team would need to continue to work at the same pace.

Reference

[1] Rosa M C, Luis V.P, Manuel M.G, Paolo R, *Authorship Attribution using Word Sequences*, Universidad Politécnica de Valencia

[2] Fung G, Mangasarian O, *The disputed Federalist Papers: SVM Feature Selection via Concave Minimization*, 14 April 2003

[3] Zhao, Y, Zobel, J, *Effective and Scalable Authorship Attribution Using Function Words,* RMIT University

[4] Eddy H. T., *The Characteristic Curves of Composition*, <http://www.jstor.org/stable/1763509>, viewed August 2010

[5] Smith, M. W. A., Recent experience and new developments of methods for the determination of authorship, *ALLC Bulletin*, 11:73–82, 1983.

[6] Hilton, J. L., On Verifying Wordprint Studies: Book of Mormon Authorship, *BYU Studies*, vol. 30, 1990.

[7] Holmes, D. I., & Forsyth, R. S., The 'Federalist' Revisited: New Directions in Authorship Attribution, *Literary and Linguistic Computing*, 10, 111-127, 1995.

[8] Stamatatos, E., Fakotakis, N. & Kokkinakis, G., Automatic Text Categorization in Terms of Genre and Author, *Computational Linguistics,* vol. 26, no. 4, pp. 471-495(25), December 2000

[9] Baayen, H., Halteren, H. V., Neijt, A. & Tweedie, F., An experiment in authorship attribution, *6th JADT*, 2002

[10] Juola, P. & Baayen, H., A controlled corpus experiment in authorship attribution by crossentropy, *Proceedings of ACH/ALLC- 2003*, 2003.

[11] Sabordo,M., Shong, C. Y., Berryman, M. J. & Abbott, D., *Who Wrote the Letter to the Hebrews? – Data Mining for Detection of Text Authorship*, SPIE vol. 5649 pp. 513 – 524, 2004.

[12] Putnins, T. J., Signoriello, D. J., Jain, S. Berryman, M. J., & Abbott, D., *Who wrote the Letter to the Hebrews? Data mining for detection of text authorship*, University of Adelaide, 2005

[13] Anderson, P. C, *The Epistle to the Hebrews and The Pauline Letter Collection*¸Harvard Theological Review, Vol. 59, No. 4 (Oct., 1966), pp. 429-438

Appendices

Appendix D:Work Breakdown Structure

WBS ID	Description	Responsible
1.	Text Authorship	Leng Tan, Jie Dong, Tien En Phua
1.1	Research on project and controversies	Leng Tan, Jie Dong, Tien En Phua
1.2	Proposal Seminar	Leng Tan, Jie Dong, Tien En Phua
1.3	Research Methods	Leng Tan, Jie Dong, Tien En Phua
1.3.1	Function Word Frequency Analysis	Tien En Phua
1.3.2	Word Recurrence Interval	Leng Tan
1.3.3	Trigram Markov Chain	Jie Dong
1.4	Stage 1 design document	Leng Tan, Jie Dong, Tien En Phua
1.5	Peer Review of Stage 1 Design	Leng Tan, Jie Dong, Tien En Phua
1.6	Software Architecture Design	Leng Tan, Jie Dong, Tien En Phua
1.7	Development of Algorithm	Leng Tan, Jie Dong, Tien En Phua
1.7.1	Function Word Frequency Analysis	Tien En Phua
1.7.2	Word recurrence Interval	Leng Tan
1.7.3	Trigram Markov Chain	Jie Dong
1.8	Progress Report	Leng Tan, Jie Dong, Tien En Phua
1.9	Test and Evaluation	Leng Tan, Jie Dong, Tien En Phua

1.9.1	Function Word Frequency Analysis	Tien En Phua
1.9.2	Word recurrence Interval	Leng Tan
1.9.3	Trigram Markov Chain	Jie Dong
1.10	SVM Implementation	Leng Tan, Jie Dong, Tien En Phua
1.10.1	Code development	Leng Tan, Jie Dong, Tien En Phua
1.10.2	Testing of develop code	Leng Tan, Jie Dong, Tien En Phua
1.11	Interim Performance Report	Leng Tan, Jie Dong, Tien En Phua
1.12	Software Modification	Leng Tan, Jie Dong, Tien En Phua
1.13	Analysis of results	Leng Tan, Jie Dong, Tien En Phua
1.14	Comparison of results	Leng Tan, Jie Dong, Tien En Phua
1.15	Implementation on controversies	Leng Tan, Jie Dong, Tien En Phua
1.16	Final seminar	Leng Tan, Jie Dong, Tien En Phua
1.17	Final Report	Leng Tan, Jie Dong, Tien En Phua
1.18	Final performance	Leng Tan, Jie Dong, Tien En Phua
1.19	Final project exhibition	Leng Tan, Jie Dong, Tien En Phua

Appendix C: Gannt Chart