Intrinsic Plagiarism Detection and Prevention System

Developed by:
Rafey Ahmad
2080-FBAS/BSCS/F10
Adnan Shah
1954-FBAS/BSCS/F10

Supervised by:
Dr. Ali Daud
Assistant Professor
Department of Computer Science & Software Engineering

Department of Computer Science & Software Engineering
Faculty of Basic and Applied Sciences
International Islamic University, Islamabad
2014
International Islamic University, Islamabad  
Faculty of Basic and Applied Sciences  
Department of Computer Science & Software Engineering  
Final Approval 

Dated: ________________

It is to certify that we have read this project report submitted by Rafey Ahmad Registration No. 2080-FBAS/BSCS/F10 and Adnan Shah Registration No. 1954-FBAS/BSCS/F10. and it is our judgment that this project is to sufficient standard to warrant its acceptance by the International Islamic University, Islamabad for the award of Degree of Bachelor of Science in Computer Sciences.

Committee

External Examiner

______________________________
Department of Computer Science & Software Engineering
International Islamic University, Islamabad

Internal Examiner

______________________________
Department of Computer Science & Software Engineering
International Islamic University, Islamabad

Supervisor

Dr. Ali Daud
Assistant Professor,
Department of Computer Science & Software Engineering
International Islamic University, Islamabad
A dissertation submitted to the
Department of Computer Science & Software Engineering,
International Islamic University, Islamabad
as a partial fulfillment of the requirements
for award of the degree of
Bachelors in Computer Sciences
Declaration

We hereby declare that this software has not been copied out from any source. It is further declared that all the information contained in this document is our own production made with the guidance of our supervisor. No portion of the work presented in this report has been submitted in support of any application for any other degree or qualification of this or any other university or institute of learning.

Rafey Ahmad
2080-FBAS/BSCS/F10

Adnan Shah
1954-FBAS/BSCS/F10
Acknowledgement

All praise to Almighty ALLAH (s.w.t), the Creator and the Maintainer of the universe, empowered us to complete this project.

We would like to thank our supervisor Dr. Ali Daud, his support and help all the way through the development of this project. He assisted us always with the tools and process of development to improve our project.

Finally we feel pride to mention the full-nourished smiling faces of our Parents and our family, the tribute goes to them.
# Project in Brief

<table>
<thead>
<tr>
<th><strong>Project Title:</strong></th>
<th>Intrinsic Plagiarism Detection and Prevention System</th>
</tr>
</thead>
</table>
| **Undertaken By:** | Rafey Ahmad  
Registration No.: 2080-FBAS/BSCS/F10  
Adnan Shah  
Registration No.: 1954-FBAS/BSCS/F10 |
| **Supervised By:** | Dr. Ali Daud  
Assistant Professor,  
Department of Computer Science & Software Engineering,  
International Islamic University, Islamabad |
| **Date Started:** | March 18, 2014 |
| **Date Completed:** | September 14, 2014 |
| **Tools Used:** | Visual Studio 2013.  
Microsoft Excel 2013.  
Microsoft Visio 2013.  
Windows Azure. |
| **OS Used:** | Microsoft Windows 8 |
Abstract

In this document we will define high-level needs and features of “Intrinsic Plagiarism Detection and Prevention System”. It will describe the needs and information of stakeholders about enhancing work, and how these needs will be fulfilled by our proposed application. The details of features of proposed application are explained below.

The existing Plagiarism detection tools are lacking the desired performance and consuming immense resources like large databases and indexing. They incur high development and maintenance costs. These systems fail when references are not given. They consume a lot of time finding the source document from which text was copied. Although they are accurate but we need a quick way to detect plagiarism within no time.

The solution is to develop a system which uses stylometry to detect plagiarism. It uses stylistic features and identifies style changes with the help of rule-based algorithms within a single document without use of external database or online tools. The key features are the performance and ease of use.

We evaluated every single feature on the plagiarism corpus and selected the best rules. Combining these best rules we get rule providing best results. We obtained 72% accuracy, 49% precision, 55% recall and 52% f measure on paragraph detection using verb count, noun count, TF-IDF and word length. On sentence detection, we obtained 75% accuracy, 43% precision, 46% recall and 45% f measure using TF-IDF.

It is challenging to develop such a system but our developed tool provides adequate accuracy on the stimulated plagiarism corpus. It detects potential plagiarism at both paragraph and sentence level. The moment you upload a document you get the plagiarism detection results within a few seconds.
# Table of contents

## Chapter 1

1 **Introduction**

1.1 Plagiarism

1.2 External Plagiarism Detection

1.3 Intrinsic Plagiarism Detection

1.4 Existing Systems

1.4.1 Word N-gram based analysis

1.4.2 Using Word-Net based semantic similarity

1.4.3 Term Occurrence Analysis

1.5 Problem Statement

1.6 Stakeholders and Interests

1.6.1 Teaching Faculty

1.6.2 Academic Institutions

1.6.3 Professional Writers

1.6.4 Students

1.7 Proposed Solution

1.8 Main Modules

## Chapter 2

2 **System Analysis**

2.1 Actors

2.1.1 Teaching Faculty

2.1.2 Academic Institutions

2.1.3 Professional Writers

2.1.4 Students

2.2 Use Case Diagram

2.3 Use Case Description in Brief

2.4 Use Case Description in Detail

2.5 System Sequence Diagram

2.6 Domain/Conceptual Model
2.7 Stylometry ........................................................................................................ 14
  2.7.1 Methods ........................................................................................................ 15
  2.7.2 Rule-based Algorithms ................................................................................ 15

2.8 Simulated Plagiarism Corpus ....................................................................... 15

2.9 Features Extraction ......................................................................................... 16
  2.9.1 Feature set ..................................................................................................... 16
  2.9.2 Significance ................................................................................................... 16

2.10 Binary classification ....................................................................................... 17

2.11 Performance Evaluation Measures ................................................................. 17
  2.11.1 Accuracy ..................................................................................................... 17
  2.11.2 Precision .................................................................................................... 18
  2.11.3 Recall .......................................................................................................... 18
  2.11.4 F measure .................................................................................................. 18

2.12 Rule Generation ............................................................................................... 19
  2.12.1 Combinations ............................................................................................. 20
  2.12.2 Rule Combinations .................................................................................... 21

Chapter 3 .............................................................................................................. 25

3 System Design ..................................................................................................... 26

3.1 Class Diagram .................................................................................................. 26

3.2 Sequence Diagram ........................................................................................... 28

3.3 Activity Diagram .............................................................................................. 29
  3.3.1 Activity Diagram (Teacher) ......................................................................... 29
  3.3.2 Activity Diagram (Academic Institution) .................................................... 30
  3.3.3 Activity Diagram (Professional Writer) ....................................................... 30
  3.3.4 Activity Diagram (Student) ........................................................................ 31
  3.3.5 Explanation .................................................................................................. 31

Chapter 4 ............................................................................................................... 32

4 Implementation ................................................................................................... 33

4.1 Complete Pseudo Code .................................................................................. 33
  4.1.1 Compute Paragraph Features Module ....................................................... 34
  4.1.2 Classify Paragraphs Module ....................................................................... 35
  4.1.3 Compute Sentence Features .................................................................... 36
  4.1.4 Classify Sentences Module ....................................................................... 37
  4.1.5 Authorship Validation Module .................................................................. 37
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Subsections</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1.6</td>
<td>Chunk mixer algorithm</td>
<td></td>
</tr>
<tr>
<td>4.1.7</td>
<td>Pseudo Code</td>
<td></td>
</tr>
<tr>
<td>4.2</td>
<td>Tools &amp; Technologies</td>
<td>4.2.1 Tools: 4.2.2 Languages, Libraries &amp; Technologies:</td>
</tr>
<tr>
<td>5</td>
<td>System Testing</td>
<td>5.1 Why software testing is essential? 5.2 Test cases</td>
</tr>
<tr>
<td>5.1.1</td>
<td>Black box testing</td>
<td>5.1.2 White box testing</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Format Verification of Uploaded Document</td>
<td>5.2.2 Feature Extraction from chunks 5.2.3 Classify Chunks 5.2.4 Validate Authorship 5.2.5 Generate Report on plagiarized passages</td>
</tr>
<tr>
<td>6</td>
<td>Results</td>
<td>6.1 Suspicious Document 6.2 Feature Extraction 6.3 Classification 6.4 Plagiarism Report</td>
</tr>
<tr>
<td>6.2.1</td>
<td>Document Features</td>
<td>6.2.2 Paragraph Features 6.2.3 Sentence Features</td>
</tr>
<tr>
<td>6.3</td>
<td>Classification</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Conclusion</td>
<td>7.1 Enhancements that can be done</td>
</tr>
<tr>
<td>8</td>
<td>Bibliography and References</td>
<td></td>
</tr>
</tbody>
</table>
Chapter 1

Introduction
1 Introduction

Our project mainly focuses on detecting potential plagiarism and authorship validation in an efficient and effective manner without use of external sources.

1.1 Plagiarism

Plagiarism is the "wrongful appropriation" and "purloining and publication" of another author's "language, thoughts, ideas, or expressions," and the representation of them as one's own original work. It is considered academic dishonesty and a breach of journalistic ethics. [1]

1.2 External Plagiarism Detection

External detection systems compare a suspicious document with a reference collection, which is a set of documents assumed to be genuine. Based on a chosen document model and predefined similarity criteria, the detection task is to retrieve all documents that contain text that is similar to a degree above a chosen threshold to text in the suspicious document. [2]

1.3 Intrinsic Plagiarism Detection

Intrinsic Plagiarism Detection Systems solely analyse the text to be evaluated without performing comparisons to external documents. This approach aims to recognize changes in the unique writing style of an author as an indicator for potential plagiarism. [2]

1.4 Existing Systems

The existing plagiarism detection systems use techniques which are further categorized as follows:

1.4.1 Word N-gram based analysis

N-grams approach has the main limitation in case someone just re-orders the words in a sentence. Also n-gram approach takes into account the spaces between words so the two words “New York” and “NewYork” will be treated differently. Also the models discussed above have Semantic sensitivity. In case the n-gram chunks are same in context but are different in words; the model will result in false negatives.
1.4.2 Using Word-Net based semantic similarity

Yurii Palkovskii et al. used Word-Net to counter for the paraphrased cases in external plagiarism detection. They performed the semantic text comparison in following steps for plagiarism detection. Given two sentences X and Y, they denoted m to be length of X, n to be length of Y and the strategy to capture semantic similarity between two sentences is given below.

- Tokenization and word stemming.
- Perform part of speech tagging.
- Word sense disambiguation.
- Building a semantic similarity relative matrix $R[m, n]$ of each pair of word senses, where $R[i, j]$ is the semantic similarity between the most appropriate sense of word at position i of X and the most appropriate sense of word at position j of Y. Thus, $R[i, j]$ is also the weight of the edge connecting from i to j.
- The match results from the previous step are combined into a single similarity value for two sentences using the following equation.

$$2 \times \text{Match}(X,Y) / X + Y$$ (1.1)

The use of word-Net for plagiarism is a time taking process as it has to do exhaustive searching within its own database for each word. Also Word-Net does not contain words relating to every field of research and education.

1.4.3 Term Occurrence Analysis

Mario Zechner et al. used the vector space model for the detection of both external and intrinsic plagiarism detection. Their aim was to speed up the search in high dimensional vector spaces at the cost of precision. They used the following three step process to achieve the task.

- First one was the vectorization of the passages of each document in the reference corpus and partitioning of the reference corpus vector space where each unique term in the reference corpus was represented as a single dimension and each sentence was a vector instead of representing a whole document as a vector,
- Second was the vectorization of the passages of a suspicious document and finding each passage’s nearest neighbor(s) in the reference corpus vector space where as the
detection of plagiarism for each suspicious document was based on its nearest neighbor list via cosine similarity threshold.

- Third one was based on post processing of plagiarized passages into single block. [3]

### 1.5 Problem Statement

- The use of word n-grams for external plagiarism detection can become unfruitful as the n-grams do not take into account semantic similarity and will fail in case someone change the words with synonyms.

- Word-Net based solutions to deal with the problem of synonyms replacement of words is time taking because word-net has to find the relationships among different words using its database of over 150,000 words.

- Efstathios Stamatatos’s method for using stop-words n-grams only could prove worthless in case someone remove or alter stop-words from plagiarized passages e.g. the sentence “Publish or Perish utility by Harzing provides the same facility of getting the author information about his published work and rank on different measuring scales” is replaced simply by “PublishOrPerish utility by Harzing provides same facility for getting author information about his published work and rank over different measuring scales”

Hence a comprehensive framework is required for the task of plagiarism detection that can be used to

- Overcome the shortcomings of n-gram approach for external plagiarism detection.
- Syntactically analyze the suspicious documents according to the sentence sense without using softwares like Word-Net.
- Fast searching of plagiarized passages in a large matrix space. [3]
1.6 Stakeholders and Interests

The main stakeholders are the teaching faculty, students, academic institutions and professional writers. Their interests in the software are elaborated below:

1.6.1 Teaching Faculty

It corresponds to the academic and teaching staff in an academic institution like college or university or research institute. The staff includes teachers, researchers, supervisors etc. They use this software to check their work and their student’s work for potential cases of plagiarism.

1.6.2 Academic Institutions

Academic Institutions include schools, colleges, universities, research centers etc. These institutions offer this tool to their teachers and faculty staff for checking plagiarism and authorship validation.

1.6.3 Professional Writers

The professional writers include book authors, blog writers, article writers, reviewers etc. They can make use of this tool for validation of their work. They can check their articles, journals etc. for plagiarism before publishing.

1.6.4 Students

The students who study in a college, school or university or if they are part of online learning courses, this software is equally useful and beneficial for them. Students are much exposed to plagiarism as they lack the linguistic performance and research skills for citation of referenced documents. They can scan their work for accidental plagiarism.

1.7 Proposed Solution

Our system will syntactically analyze the word windows in order to address the above problems that may occur while using n-grams profile. Replacing words with their respective POS tags will allow us to not to be dependent on the software like Word-Net and hence attaining the target of rapid searching of plagiarized chunks within a matrix space. [3]
1.8 Main Modules

The system covers the following modules:

- Corpus Loader
- Stylometric features extractor
- Outlier Detector Model
- Authorship Validator
- Plagiarism Report Generator
Chapter 2

System Analysis
2 System Analysis

This Section defines all the actors and use case in my application. The use case diagrams are also drawn. The sequence of the use cases is shown with the help of Sequence diagrams. The use case Scenarios are included so that the user could better understand the ongoing data flow and its control flow.

2.1 Actors

An Actor is an external agent who interacts with the system and causes an event that initiates the corresponding use case. An Actor is a user of the system; user can mean a human user, a machine, or even another system or subsystem in the model. Anything that interacts with the system from the outside or system boundary is termed an Actor. Actors are typically associated with use cases.

Actors can use the system through a graphical user interface, through a batch interface or through some other media. An Actor's interaction with a Use Case is documented in a Use Case scenario, which details the functions a system must provide to satisfy the user requirements.

In Intrinsic Plagiarism Detection System there are four actors.

- The Teaching Faculty
- Academic Institutions
- Professional Writers
- The Students.

2.1.1 Teaching Faculty

It corresponds to the academic and teaching staff in an academic institution like college or university or research institute. The staff includes teachers, researchers, supervisors etc. They use this software to check their work and their student's work for potential cases of plagiarism.

2.1.2 Academic Institutions

Academic Institutions include schools, colleges, universities, research centers etc. These institutions offer this tool to their teachers and faculty staff for checking plagiarism and authorship validation.
2.1.3 Professional Writers

The professional writers include book authors, blog writers, article writers, reviewers etc. They can make use of this tool for validation of their work. They can check their articles, journals etc. for plagiarism before publishing.

2.1.4 Students

The students who study in a college, school or university or if they are part of online learning courses, this software is equally useful and beneficial for them. Students are much exposed to plagiarism as they lack the linguistic performance and research skills for citation of referenced documents. They can scan their work for accidental plagiarism.

2.2 Use Case Diagram

A use case is a sequence of actions that provide a measurable value to an actor. Another way to look at it is a use case describes a way in which a real-world actor interacts with the system. In a system use case you include high-level implementation decisions. System use cases can be written in both an informal manner and a formal manner.

Use case Diagrams are created to visualize the relationship between actors and use cases. A use case diagram is also used to capture the system functionality as seen by the user.
Diagram

The generic use case diagram is given below:

2.1 Use Case Diagram (Generic)
2.3 Use Case Description in Brief

Anyone can check their document for plagiarism by uploading it on the server. The server generates plagiarism report in seconds with high performance classification.

2.4 Use Case Description in Detail

Table 2.1
Use Case- Check Document for Plagiarism

<table>
<thead>
<tr>
<th>Use Case ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>UC-01</td>
<td>Check Document for Plagiarism</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Primary Actor</th>
<th>Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teacher, Professional Writer, Academic Institutions, Student</td>
<td>Get Report for plagiarized passages in the uploaded document.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Precondition</th>
<th>Post Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>The document should be in text format.</td>
<td>Connection is established and detailed report is generated.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Main Success Scenario</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Actor opens main page</td>
<td>2. Actor clicks the choose file button.</td>
</tr>
<tr>
<td>3. Actor clicks on Process button.</td>
<td>4. The system will connect to the Server. The detection tool will generate report for plagiarized passages.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Failure Case Scenario</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. In case of failure, user is directed to upload the document again.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Scenario Extensions</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. When Actor uploads file in wrong format error is generated.</td>
<td>2. User cannot send GET request to the Process page.</td>
</tr>
</tbody>
</table>


2.5 System Sequence Diagram

A System Sequence Diagram (SSD) is a fast and easily created artifact that illustrates input and output events related to the systems under discussion.

The UML does not define something called a "system" sequence diagram but simply a "sequence diagram." Sequence diagrams illustrate interactions in a kind of fence format, in which each new object is added to the right. UML tools usually emphasize sequence diagrams, because of their greater notational power. Thus, tool support is better and more notation options are available. Also, it is easier to see the call-flow sequence with sequence diagrams simply read top to bottom.

A sequence diagram is an interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.
Following is sequence diagram of the software.

2.5 Sequence Diagram
2.6 Domain/Conceptual Model

A Domain Model is a high-level conceptual model, defining physical and abstract objects in an area of interest to the Project. It can be used to document relationships between and responsibilities of conceptual classes (that is, classes that represent the concept of a group of things rather than Classes that define a programming object). It is also useful for defining the terms of a domain.

2.7 Stylometry

Stylometry is the application of the study of linguistic style, usually to written language, but it has successfully been applied to music and to fine-art paintings as well.

Stylometry is often used to attribute authorship to anonymous or disputed documents. It has legal as well as academic and literary applications, ranging from the question of the authorship of Shakespeare's works to forensic linguistics.
2.7.1 Methods

Modern stylometry draws heavily on the aid of computers for statistical analysis, artificial intelligence and access to the growing corpus of texts available via the Internet.

- Writer Invariant
- Neural Networks
- Rule-based algorithms
- Genetic Algorithms
- Rare pairs

2.7.2 Rule-based Algorithms

The rule based algorithm is another artificial intelligence technique used in stylometry. This involves a method that starts out with a set of rules. An example rule might be, "If but appears more than 1.7 times in every thousand words, then the text is author X". The program is presented with text and uses the rules to determine authorship. The rules are tested against a set of known texts and each rule is given a fitness score. The 50 rules with the lowest scores are thrown out. The remaining 50 rules are given small changes and 50 new rules are introduced. This is repeated until the evolved rules correctly attribute the texts. [4]

2.8 Simulated Plagiarism Corpus

We have created a plagiarism corpus which contains almost 2000 cases of simulated plagiarism from set of 100 research thesis of Brown University UK. The cases are generated by our chunk mixer algorithm. This corpus is created for testing our rule-based algorithms for stylometric analysis.

This algorithm chooses a document randomly and retrieves a chunk from that document at random position and mixes it in destination document at random position in it.
2.9 Features Extraction

When the input data to an algorithm is too large to be processed and it is suspected to be notoriously redundant (e.g. the same measurement in both feet and meters) then the input data will be transformed into a reduced representation set of features (also named features vector). Transforming the input data into the set of features is called feature extraction. If the features extracted are carefully chosen it is expected that the features set will extract the relevant information from the input data in order to perform the desired task using this reduced representation instead of the full size input. [5]

2.9.1 Feature set

The feature set used for style retrieval consists of 13 stylometric features:

- Adjective Count
- Noun Count
- Pronoun Count
- Proper Noun Count
- Verb Count
- Adverb Count
- Sentence Length
- Stop words Count
- Word Length
- Syllable Count
- Paragraph Length
- Punctuation Count
- TF-IDF (Term Frequency- Inverse Document Frequency)

2.9.2 Significance

The feature set stated above is pretty useful in stylometric analysis. In plagiarism detection, TF-IDF and POS tags are relatively more significant. Text difficulty level can be recognized by sentence length and syllable count and word count. Stop words classify the writing style very well. Punctuation count can classify certain authors as they use different number of punctuations.
2.10 Binary classification

Binary or binomial classification is the task of classifying the elements of a given set into two groups on the basis of a classification rule. Some typical binary classification tasks are:

- medical testing to determine if a patient has certain disease or not – the classification property is the presence of the disease;
- quality control in factories; i.e. deciding if a new product is good enough to be sold, or if it should be discarded – the classification property is being good enough;
- Information retrieval, namely deciding whether a page or an article should be in the result set of a search or not – the classification property is the relevance of the article, or the usefulness to the user. [6]

2.11 Performance Evaluation Measures

These measures are used to evaluate the performance of our detection tool. We have used four performance evaluation measures for our system.

- Accuracy
- Precision
- Recall
- F measure

2.11.1 Accuracy

Accuracy is also used as a statistical measure of how well a binary classification test correctly identifies or excludes a condition.

That is, the accuracy is the proportion of true results (both true positives and true negatives) in the population. To make the context clear by the semantics, it is often referred to as the "Rand Accuracy". It is a parameter of the test. [7]

\[
\text{Accuracy} = \frac{\text{true positives count} + \text{true negatives count}}{\text{true positives} + \text{false positives} + \text{true negatives} + \text{false negatives}} \quad \cdots (2.1)
\]
2.11.2 Precision

In pattern recognition and information retrieval with binary classification, precision (also called positive predictive value) is the fraction of retrieved instances that are relevant.

In a classification task, the precision for a class is the number of true positives (i.e. the number of items correctly labeled as belonging to the positive class) divided by the total number of elements labeled as belonging to the positive class (i.e. the sum of true positives and false positives, which are items incorrectly labeled as belonging to the class).

Precision or positive predictive value is defined as the proportion of the true positives against all the positive results (both true positives and false positives). [8]

\[
Precision = \frac{\text{number of true positives}}{\text{true positives} + \text{false positives}} \quad \cdots (2.2)
\]

2.11.3 Recall

Recall (also known as sensitivity) is the fraction of relevant instances that are retrieved.

Recall in this context is defined as the number of true positives divided by the total number of elements that actually belong to the positive class (i.e. the sum of true positives and false negatives, which are items which were not labeled as belonging to the positive class but should have been). [8]

\[
Recall = \frac{\text{number of true positives}}{\text{true positives} + \text{false negatives}} \quad \cdots (2.3)
\]

2.11.4 F measure

In statistical analysis of binary classification, the F1 score (also F-score or F-measure) is a measure of a test's accuracy. It considers both the precision p and the recall r of the test to compute the score: p is the number of correct results divided by the number of all returned results and r is the number of correct results divided by the number of results that should have been returned. The F1 score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0.

The traditional F-measure or balanced F-score (F1 score) is the harmonic mean of precision and recall. [9]

\[
F \text{ measure} = 2 \cdot \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \quad \cdots (2.4)
\]
2.12 Rule Generation

The pattern for generating the rules is as follows:

_{If} (chunk feature value > document feature value + classifying factor) OR

(chunk feature value < document feature value - classifying factor)
_{Then} chunk is plagiarized
_{Else} chunk is not plagiarized

Following this pattern, a large number of rules were generated for classification using feature set of 13 features and varying values of classifying factor and different chunk sizes.

We selected top 25 rules which best fit the corpus data. The accuracy, precision, recall and f measure given by these rules is in the following tables:

**Table 2.2**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Sentence Length</th>
<th>Stop Word Count</th>
<th>Word Length</th>
<th>Syllable Count</th>
<th>Paragraph Length</th>
<th>Punctuation Count</th>
<th>TF-IDF</th>
</tr>
</thead>
<tbody>
<tr>
<td>(+/-) factor</td>
<td>5.400</td>
<td>5.0</td>
<td>0.370</td>
<td>0.130</td>
<td>1.900</td>
<td>2.25</td>
<td>6.68</td>
</tr>
<tr>
<td>Accuracy</td>
<td>63.53</td>
<td>65.13</td>
<td>63.03</td>
<td>61.43</td>
<td>65.23</td>
<td>62.93</td>
<td>71.22</td>
</tr>
<tr>
<td>Precision</td>
<td>33.09</td>
<td>35.66</td>
<td>32.38</td>
<td>31.97</td>
<td>35.68</td>
<td>32.87</td>
<td>47.21</td>
</tr>
<tr>
<td>Recall</td>
<td>33.94</td>
<td>35.79</td>
<td>33.57</td>
<td>37.63</td>
<td>35.42</td>
<td>35.42</td>
<td>53.13</td>
</tr>
<tr>
<td>F measure</td>
<td>33.515</td>
<td>35.72</td>
<td>32.97</td>
<td>34.57</td>
<td>35.55</td>
<td>34.10</td>
<td>50.00</td>
</tr>
</tbody>
</table>

**Table 2.3**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Sentence Length</th>
<th>Stop Word Count</th>
<th>Word Length</th>
<th>Syllable Count</th>
<th>Punctuation Count</th>
<th>TF-IDF</th>
</tr>
</thead>
<tbody>
<tr>
<td>(+/-) factor</td>
<td>9.9</td>
<td>9.26</td>
<td>0.675</td>
<td>0.252</td>
<td>4.3</td>
<td>12.48</td>
</tr>
<tr>
<td>Accuracy</td>
<td>65.05</td>
<td>61.50</td>
<td>65.15</td>
<td>64.62</td>
<td>62.94</td>
<td>74.55</td>
</tr>
<tr>
<td>Precision</td>
<td>25.72</td>
<td>24.01</td>
<td>26.10</td>
<td>24.96</td>
<td>24.68</td>
<td>43.28</td>
</tr>
<tr>
<td>Recall</td>
<td>27.35</td>
<td>33.72</td>
<td>27.06</td>
<td>26.62</td>
<td>32.45</td>
<td>46.27</td>
</tr>
<tr>
<td>F measure</td>
<td>26.50</td>
<td>28.05</td>
<td>26.57</td>
<td>25.76</td>
<td>28.04</td>
<td>44.72</td>
</tr>
</tbody>
</table>
Table 2.4
Paragraph Features Best Performance (POS Tags)

<table>
<thead>
<tr>
<th>Feature</th>
<th>Noun Count</th>
<th>Proper Noun Count</th>
<th>Pronoun Count</th>
<th>Verb Count</th>
<th>Adverb Count</th>
<th>Adjective Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>(+/-) factor</td>
<td>4.2</td>
<td>0.18</td>
<td>1.76</td>
<td>3.6</td>
<td>1.95</td>
<td>2.5</td>
</tr>
<tr>
<td>Accuracy</td>
<td>63.33</td>
<td>60.73</td>
<td>63.03</td>
<td>65.43</td>
<td>60.73</td>
<td>60.73</td>
</tr>
<tr>
<td>Precision</td>
<td>33.44</td>
<td>28.96</td>
<td>32.00</td>
<td>36.36</td>
<td>31.40</td>
<td>28.67</td>
</tr>
<tr>
<td>Recall</td>
<td>35.79</td>
<td>30.99</td>
<td>32.47</td>
<td>36.90</td>
<td>38.00</td>
<td>30.25</td>
</tr>
<tr>
<td>F measure</td>
<td>34.53</td>
<td>29.94</td>
<td>32.23</td>
<td>36.63</td>
<td>34.39</td>
<td>29.53</td>
</tr>
</tbody>
</table>

Table 2.5
Sentence Features Best Performance (POS Tags)

<table>
<thead>
<tr>
<th>Feature</th>
<th>Noun Count</th>
<th>Proper Noun Count</th>
<th>Pronoun Count</th>
<th>Verb Count</th>
<th>Adverb Count</th>
<th>Adjective Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>(+/-) factor</td>
<td>9.0</td>
<td>0.21</td>
<td>2.77</td>
<td>6.83</td>
<td>3.7</td>
<td>5.76</td>
</tr>
<tr>
<td>Accuracy</td>
<td>66.68</td>
<td>64.76</td>
<td>63.24</td>
<td>62.70</td>
<td>63.61</td>
<td>63.91</td>
</tr>
<tr>
<td>Precision</td>
<td>26.64</td>
<td>23.07</td>
<td>26.49</td>
<td>22.92</td>
<td>23.46</td>
<td>22.94</td>
</tr>
<tr>
<td>Recall</td>
<td>28.36</td>
<td>25.00</td>
<td>36.72</td>
<td>28.63</td>
<td>28.09</td>
<td>26.36</td>
</tr>
<tr>
<td>F measure</td>
<td>27.47</td>
<td>23.99</td>
<td>30.78</td>
<td>25.46</td>
<td>25.56</td>
<td>24.53</td>
</tr>
</tbody>
</table>

2.12.1 Combinations

In mathematics, a combination is a way of selecting members from a grouping, such that (unlike permutations) the order of selection does not matter. In smaller cases it is possible to count the number of combinations. For example given three fruits, say an apple, an orange and a pear, there are three combinations of two that can be drawn from this set: an apple and a pear; an apple and an orange; or a pear and an orange. More formally, a \( k \)-combination of a set \( S \) is a subset of \( k \) distinct elements of \( S \). If the set has \( n \) elements, the number of \( k \)-combinations is equal to the binomial coefficient

\[
\binom{n}{k} = \frac{n(n-1)\ldots(n-k+1)}{k(k-1)\ldots1},
\]

… (2.5)
which can be written using factorials as \( \frac{n!}{k!(n-k)!} \) whenever \( k \leq n \), and which is zero when \( k > n \). The set of all \( k \)-combinations of a set \( S \) is sometimes denoted by \( \binom{S}{k} \).

Combinations refer to the combination of \( n \) things taken \( k \) at a time without repetition. To refer to combinations in which repetition is allowed, the terms \( k \)-selection, \( k \)-multiset, or \( k \)-combination with repetition are often used. If, in the above example, it was possible to have two of any one kind of fruit there would be 3 more 2-selections: one with two apples, one with two oranges, and one with two pears.

Although the set of three fruits was small enough to write a complete list of combinations, with large sets this becomes impractical. For example, a poker hand can be described as a 5-combination (\( k = 5 \)) of cards from a 52 card deck (\( n = 52 \)). The 5 cards of the hand are all distinct, and the order of cards in the hand does not matter. There are 2,598,960 such combinations, and the chance of drawing any one hand at random is \( \frac{1}{2,598,960} \).

### 2.12.2 Rule Combinations

The top 21 rules extracted from the above process are then combined using the following pattern:

**If** (rule1) OR (rule2)  
**Then** chunk is plagiarized  
**Else** chunk is not plagiarized

**If** (rule1) OR (rule2) OR (rule3)  
**Then** chunk is plagiarized  
**Else** chunk is not plagiarized

**If** (rule1) OR (rule2) OR
Intrinsic Plagiarism Detection System

Chapter 2

System Analysis

(rule3) OR
(rule4)

Then chunk is plagiarized
Else chunk is not plagiarized

If (rule1) OR
(rule2) OR
(rule3) OR
(rule4) OR
(rule5)

Then chunk is plagiarized
Else chunk is not plagiarized

If (rule1) OR
(rule2) OR
(rule3) OR
(rule4) OR
(rule5) OR
(rule6)

Then chunk is plagiarized
Else chunk is not plagiarized

If (rule1) OR
(rule2) OR
(rule3) OR
(rule4) OR
(rule5) OR
(rule6) OR
(rule 7)

Then chunk is plagiarized
Else chunk is not plagiarized

If (rule1) OR
(rule2) OR
(rule3) OR
(rule 4) OR
(rule 5) OR
(rule 6) OR
(rule 7) OR
(rule 8) OR
...
Then chunk is plagiarized
Else chunk is not plagiarized

Following this pattern 8178 + 4083 = 12261 rule combinations were generated and tested for different values of classifying factors.

Out of these thousands of rules we selected two rules with best accuracy, precision, recall and f measure values which is as follows:

**IF** paragraph noun count > (document noun count + 6.71497005988024) **OR**
paragraph noun count < (document noun count - 6.71497005988024) **OR**
paragraph verb count > (document verb count + 5.755688622754491) **OR**
paragraph verb count < (document verb count - 5.755688622754491) **OR**
paragraph word length > (document word length +0.5915568862275449) **OR**
paragraph word length < (document word length – 0.5915568862275449) **OR**
paragraph TF-IDF > (document TF-IDF + 10.68) **OR**
paragraph TF-IDF < (document TF-IDF – 10.68)
Then paragraph is plagiarized
Else paragraph is not plagiarized

The above rule gives performance measures as follows:

**Table 2.6**

Performance of Selected Rule for Paragraph Classification

<table>
<thead>
<tr>
<th>Performance Measure</th>
<th>Percentage (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>72.128</td>
</tr>
<tr>
<td>Precision</td>
<td>48.693</td>
</tr>
<tr>
<td>Recall</td>
<td>54.982</td>
</tr>
<tr>
<td>F measure</td>
<td>51.646</td>
</tr>
</tbody>
</table>
Intrinsic Plagiarism Detection System

**Chapter 2**

*System Analysis*

*IF* sentence TF-IDF > (document TF-IDF + 12.48) *OR*

sentence TF-IDF < (document TF-IDF - 12.48)

*Then* sentence is plagiarized

*Else* sentence is not plagiarized

The above rule gives performance measures as follows:

**Table 2.7**

Performance of Selected Rule for Sentence Classification

<table>
<thead>
<tr>
<th>Performance Measure</th>
<th>Percentage (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>74.55</td>
</tr>
<tr>
<td>Precision</td>
<td>43.28</td>
</tr>
<tr>
<td>Recall</td>
<td>46.27</td>
</tr>
<tr>
<td>F measure</td>
<td>44.72</td>
</tr>
</tbody>
</table>
Chapter 3
System Design
3 System Design

Design is a process that uses the product of analysis to produce a specification for implementing a system. Design is the logical description of how a system will work.

Design emphasizes a conceptual solution that fulfills the requirements, rather than its implementation. For example, a description of a database schema and software objects. Design ideas often exclude low-level or "obvious" details obvious to the intended consumers.

Ultimately, designs can be implemented, and the implementation (such as code) expresses the true and complete realized design. The term is best qualified, as in object-oriented design or database design.

3.1 Class Diagram

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

Class or structural diagrams define the basic building blocks of a model. They are used for static object modeling, describing what attributes and behavior it has rather than detailing the methods for achieving operations.
3.1 Class Diagram
3.2 Sequence Diagram
3.3 Activity Diagram

An activity diagram is a simple and intuitive illustration of what happens in a workflow, what activities can be done in parallel, and whether there are alternative paths through the workflow. Activity diagrams as defined in the Unified Modeling Language are derived from various techniques to visually illustrate workflows.

3.3.1 Activity Diagram (Teacher)

![Activity Diagram (Teacher)]
3.3.2 Activity Diagram (Academic Institution)

3.4 Activity Diagram (Academic Institution)

3.3.3 Activity Diagram (Professional Writer)

3.5 Activity Diagram (Professional Writer)
3.3.4 Activity Diagram (Student)

3.6 Activity Diagram (Student)

3.3.5 Explanation

The above activity diagram explains the various activities involved in the workflow.
When the user uploads the document the upload document activity is started. If document is successfully uploaded, the extract features activity is started.
If there is an error, the user remains in the previous activity until the operation is successful.
In the extract features activity the features are extracted by breaking the document into chunks and calculating the feature values.
If features are extracted, the next activity is started which classifies chunks.
If plagiarized chunks are found above a certain percentage, the generate report activity is started which generates plagiarism report and then another activity is started which displays the report to the user.
If plagiarized chunks are not found, the authorship validator activity is started which validates authorship by matching the extracted features from database of features of various authors. If features are matched to a different author other than the one who claimed, authorship is compromised. Then a detailed report is generated about authorship and sent back to user.
Chapter 4

Implementation
4 Implementation

Implementation (software) perspective describes software implementations in a particular language or technology (such as Python). Implementation means programming and building the system, not deploying it.

In the implementation phase, the developer builds the components either from scratch or by composition given the architecture document from the design phase and the requirement document from the analysis phase. The architecture document should give guidance. Sometimes, this guidance is found in the requirement document. The implementation phase deals with issues of quality, performance and debugging. The end deliverable of implementation phase is the product itself.

4.1 Complete Pseudo Code

SET path to uploads folder path
READ files from path into PlaintextCorpusReader
SET corpus to PlaintextCorpusReader
READ stop words from corpus module of NLTK
SET stopwords to stop words
READ dictionary from cmudict module
SET dict to dictionary
READ first 5000 tagged sentences from Treebank corpus
SET tagged_sents to tagged sentences
CALL Train Unigram Tagger with tagged_sents returning trained tagger
CALL compute_features_para
CALL classify_chunks_para with list of classifying factors
CALL compute_features_sent
CALL classify_chunks_sent with list of classifying factors
4.1.1 Compute Paragraph Features Module

READ file names from corpus reader

FOR file in files
    INITIALIZE document feature variables
    GET words from document
    GET most frequent words which are 24% of total words
    GET least frequent words which are 24% of total words
    FOR each paragraph in file
        INITIALIZE paragraph feature variables
        GET words from paragraph
        GET most frequent words which are 66% of total words
        GET least frequent words which are 66% of total words
        GET words from paragraph which are the least frequent words in document
        DIVIDE these number of words by number of words in paragraph
        FOR each sentence in paragraph
            FOR each word in sentence
                GET syllables from dictionary for word
                ADD number of syllables in syllable count
                IF word is a stop word
                    ADD 1 to stop word count
                TAG sentence with trained tagger
                FOR each tuple in tagged sentence
                    CASE tuple of
                        Noun: ADD 1 to noun count
                        Proper Noun: ADD 1 to proper noun count
                        Pronoun: ADD 1 to Pronoun count
                        Verb: ADD 1 to Verb count
                        Adverb: ADD 1 to adverb count
                        Adjective: ADD 1 to adjective count
                GET number of words in paragraph
                ADD paragraph features to document features
            DIVIDE syllable count by number of words in paragraph
GET number of characters in paragraph
DIVIDE character count by number of words in paragraph
CALCULATE percentage of all counting features relative to paragraph
CALCULATE document features
CALCULATE percentage of all counting features relative to document

4.1.2 Classify Paragraphs Module

OPEN file at specified path for saving results
GET number of paragraphs in a document
SET factor1 to 6.71497005988024
SET factor2 to 5.755688622754491
SET factor3 to 0.5915568862275449
SET factor4 to 10.68
FOR each paragraph in document
    IF paragraph noun count > (document noun count + factor1)
    OR paragraph noun count < (document noun count - factor1)
    OR paragraph verb count > (document verb count + factor2)
    OR paragraph verb count < (document verb count - factor2)
    OR paragraph word length > (document word length + factor3)
    OR paragraph word length < (document word length - factor3)
    OR paragraph TF-IDF > (document TF-IDF + factor4)
    OR paragraph TF-IDF < (document TF-IDF - factor4)
        WRITE plagiarized to file
        ADD number of words to detected words
    ELSE
        WRITE original to file
    DIVIDE number of detected words by total number of words
    IF percentage < 5
        CALL authorship validator with document and chunk size
4.1.3 Compute Sentence Features

READ file names from corpus reader

FOR file in files
  INITIALIZE document feature variables
  GET words from document
  GET most frequent words which are 33% of total words
  GET least frequent words which are 33% of total words
  FOR each sentence in file
    INITIALIZE sentence feature variables
    GET words from sentence
    GET most frequent words which are 65% of total words
    GET least frequent words which are 65% of total words
    GET words from sentence which are the least frequent words in document
    DIVIDE these number of words by number of words in sentence
    FOR each word in sentence
      GET syllables from dictionary for word
      ADD number of syllables in syllable count
      IF word is a stop word
        ADD 1 to stop word count
      TAG sentence with trained tagger
      FOR each tuple in tagged sentence
        CASE tuple of
          Noun: ADD 1 to noun count
          Proper Noun: ADD 1 to proper noun count
          Pronoun: ADD 1 to Pronoun count
          Verb: ADD 1 to Verb count
          Adverb: ADD 1 to adverb count
          Adjective: ADD 1 to adjective count
        GET number of words in sentence
        ADD sentence features to document features
        DIVIDE syllable count by number of words in sentence
        GET number of characters in sentence
DIVIDE character count by number of words in sentence
CALCULATE percentage of all counting features relative to sentence
CALCULATE document features
CALCULATE percentage of all counting features relative to document

4.1.4 Classify Sentences Module

OPEN file at specified path for saving results
GET number of sentences in a document
SET factor to 12.48
FOR each sentence in document
  IF sentence TF-IDF > (document TF-IDF + factor)
    OR sentence TF-IDF < (document TF-IDF - factor)
      WRITE plagiarized to file
      ADD number of words to detected words
    ELSE
      WRITE original to file
  DIVIDE number of detected words by total number of words
IF percentage < 5
  CALL authorship validator with document and chunk size

4.1.5 Authorship Validation Module

OPEN connection to author styles db
GET author metadata and feature set from db
GET difference of document features and author features
GET author with minimum feature difference
IF author == claiming author
  WRITE authorship validated to file
ELSE
  WRITE authorship compromised and author name to file
4.1.6 Chunk mixer algorithm

This algorithm chooses a document randomly and retrieves a chunk from that document at random position and mixes it in destination document at random position in it.

4.1.7 Pseudo Code

SET corpus to PlainTextCorpusReader with folder path
READ file ids from corpus
SET docs to file ids
CALL mixParas
CALL mixSents
Public mixParas ()
{
    FOR each file in corpus
        SET logtext to empty string
        READ paragraphs from file
        SET file_paras to paragraphs
        FOR each number from 1 to number of paragraphs*0.25
            READ a document at random position from corpus
            READ paragraphs from document
            SET doc_paras to paragraphs
            GET paragraph at random position from doc_paras
            SET para to paragraph
            SET paragraph at random position to para
        WRITE logtext to log file
        WRITE paragraphs to new file
    }
Public mixSents ()
{
    FOR each file in corpus
        SET logtext to empty string
        READ sentences from file
        SET file_sents to sentences
        FOR each number from 1 to number of sentences*0.25
            READ a document at random position from corpus
            READ sentences from document
            SET doc_sents to sentences
            GET sentence at random position from doc_sents
            SET sent to sentence
            SET sentence at random position to sent
        WRITE logtext to log file
        WRITE sentences to new file
    }
4.2 Tools & Technologies

Following are the tools and technologies used to develop this project:

4.2.1 Tools:

- Visual Studio 2013.
- Microsoft Excel 2013.
- Microsoft Visio 2013.
- Windows Azure.

4.2.2 Languages, Libraries & Technologies:

- Python
- NLTK
- Matplotlib
- Numpy
- Treebank corpus
- Unigram Tagger
- ITER tools
- C#
- ASP.NET MVC
- Bootstrap
- jQuery
- HTML 5
- CSS 3
- Microsoft Windows 8
Chapter 5

System Testing
5 System Testing

Software testing is the process of evaluation a software item to detect differences between given input and expected output. Also to assess the features of a software item. Testing assesses the quality of the product. Software testing is a process that should be done during the development process. In other words software testing is a verification and validation process.

Testing is an essential step in the development of a reliable software system. There are different types of testing in which related activities are performed. By examining the system we find defects and their root causes and then fix those defects to make the system functional. Sometimes we execute the system and try to make it fail by entering false inputs. Testing uncovers the hidden defects in the system and let the developers to fix these defects before the product is released.

5.1 Why software testing is essential?

First to make sure the program runs properly without crashes or errors. Also it is essential to check to see if there are security issues. One of the main aims of testing is authentication and validation. Reliability of the software is an important factor that is determined by the testing. With help of testing in software development, any product can be transformed into a strong and consistent product.

Review and inspection of code and software documents can uncover the errors but they are not sufficient and an alternate to the automated testing of the software system. The final ready program is executed for the purpose of finding and removing errors. It is important before the software is given to the customer for use. In order to find the defects, test cases must be carried out in a systematic and disciplined way using the suitable test design techniques.

There are different types of testing. But the two basic are under given. Different techniques of testing are not alternate for each other but every one of them has its importance in the testing of the whole system.
5.1.1 Black box testing

Black box testing is a testing technique that ignores the internal mechanism of the system and focuses on the output generated against any input and execution of the system. It is also called functional testing.

5.1.2 White box testing

White box testing is a testing technique that takes into account the internal mechanism of a system. It is also called structural testing and glass box testing.

5.2 Test cases

5.2.1 Format Verification of Uploaded Document

<table>
<thead>
<tr>
<th>Test Case ID</th>
<th>TC 001</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Engineer</td>
<td>Rafey Ahmad</td>
</tr>
<tr>
<td>Functional Area</td>
<td>Upload</td>
</tr>
<tr>
<td>Test Name</td>
<td>Format Verification of Uploaded Document</td>
</tr>
<tr>
<td>Objective</td>
<td>The purpose of this test case is to verify that system is allowing actor to upload a file with restricted and supported formats.</td>
</tr>
<tr>
<td>Module Environment</td>
<td>The system is running and displaying file upload form.</td>
</tr>
</tbody>
</table>
| Steps To Perform | 1. Upload File  
2. Click Process button |
| Expected Result | The file is uploaded and report page is being loaded. |
| Test Result | ☑ Passed ☐ Failed |
5.2.2 Feature Extraction from chunks

Table 5.2
Test Case- Feature Extraction from chunks

<table>
<thead>
<tr>
<th>Test Case ID</th>
<th>TC 002</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Engineer</td>
<td>Rafey Ahmad</td>
</tr>
<tr>
<td>Functional Area</td>
<td>Plagiarism Detection</td>
</tr>
<tr>
<td>Test Name</td>
<td>Feature Extraction from chunks</td>
</tr>
<tr>
<td>Objective</td>
<td>The purpose of this test case is to extract features from chunks of document.</td>
</tr>
<tr>
<td>Module Environment</td>
<td>The system is processing the uploaded file.</td>
</tr>
<tr>
<td>Expected Result</td>
<td>The feature set is extracted and passed to chunk classifier.</td>
</tr>
<tr>
<td>Test Result</td>
<td>☑ Passed □ Failed</td>
</tr>
</tbody>
</table>

5.2.3 Classify Chunks

Table 5.3
Test Case- Classify Chunks based on feature values and classification rule.

<table>
<thead>
<tr>
<th>Test Case ID</th>
<th>TC 003</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Engineer</td>
<td>Rafey Ahmad</td>
</tr>
<tr>
<td>Functional Area</td>
<td>Plagiarism Detection</td>
</tr>
<tr>
<td>Test Name</td>
<td>Classify Chunks based on feature values and classification rule.</td>
</tr>
<tr>
<td>Objective</td>
<td>The purpose of this test case is to classify chunks on basis of their feature values according to classification rule.</td>
</tr>
<tr>
<td>Module Environment</td>
<td>The system is processing the uploaded file.</td>
</tr>
<tr>
<td>Expected Result</td>
<td>The chunks are classified as plagiarized and originals.</td>
</tr>
<tr>
<td>Test Result</td>
<td>☑ Passed □ Failed</td>
</tr>
</tbody>
</table>
5.2.4 Validate Authorship

Table 5.4
Test Case- Validate Authorship

<table>
<thead>
<tr>
<th>Test Case ID</th>
<th>TC 004</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Engineer</td>
<td>Rafey Ahmad</td>
</tr>
<tr>
<td>Functional Area</td>
<td>Plagiarism Detection</td>
</tr>
<tr>
<td>Test Name</td>
<td>Validate Authorship</td>
</tr>
<tr>
<td>Objective</td>
<td>The purpose of this test case is to validate authorship by comparing document features with author features stored in DB.</td>
</tr>
<tr>
<td>Module Environment</td>
<td>The system is processing the uploaded file.</td>
</tr>
<tr>
<td>Expected Result</td>
<td>The authorship is validated and result is generated.</td>
</tr>
<tr>
<td>Test Result</td>
<td>☑ Passed ☐ Failed</td>
</tr>
</tbody>
</table>

5.2.5 Generate Report on plagiarized passages.

Table 5.5
Test Case- Generate Report on plagiarized passages.

<table>
<thead>
<tr>
<th>Test Case ID</th>
<th>TC 005</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Engineer</td>
<td>Rafey Ahmad</td>
</tr>
<tr>
<td>Functional Area</td>
<td>Plagiarism Detection</td>
</tr>
<tr>
<td>Test Name</td>
<td>Generate Report on plagiarized passages.</td>
</tr>
<tr>
<td>Objective</td>
<td>The purpose of this test case is to generate report on plagiarized passages and authorship validation.</td>
</tr>
<tr>
<td>Module Environment</td>
<td>The system is processing the uploaded file.</td>
</tr>
<tr>
<td>Expected Result</td>
<td>The report is generated and displayed on the results page on the client’s browser.</td>
</tr>
<tr>
<td>Test Result</td>
<td>☑ Passed ☐ Failed</td>
</tr>
</tbody>
</table>
Chapter 6

Results
6 Results

This chapter focusses on the in depth detail of actual working of our plagiarism detection tool by explaining with sample suspicious document. The sample document is as follows:

6.1 Suspicious Document

Users have long relied on the Internet to publish and share data with each other. Traditionally, a shared data repository is written and organized by only a single author. There are many examples for single-writer repositories. Blogs are generally managed by one user rather than a group of users. Flickr allows individuals to upload, organize, and share their own photos.

With the rise of web 2.0, data sharing has become more and more collaborative such that multiple writers can jointly write and organize the content in the repositories. This trend is evident in the emergence of online collaborations, such as Wikipedia and Google groups. Wikipedia is a popular online encyclopedia that allows users to share their knowledge by modifying the encyclopedic entries. Google groups is a repository where users can post messages and share files. As a concrete example, consider a group of friends wanting to share the photos that they take at gatherings.

At each party, multiple friends take pictures with their own cameras, which yields multiple sets of photos for the same party. It would be useful if all the friends were able to collaboratively organize the photos in a shared photo repository. The state of the art approach to build a consistent and reliable shared repository is to rely on a central site managed by a third party. However, centralized solutions are undesirable sometimes because of privacy concerns and censorship, which are problems that can be alleviated by switching to decentralized solutions. This thesis contributes the design and implementation of a decentralized multiple writer data repository that can support a variety of collaborative data sharing applications, such as repositories for sharing music and movies, bulletin boards, etc.

The current practice of data sharing is for users to upload data onto a central site, such as Wikipedia and Google. Centralized solutions have many benefits. First, a central site can replicate the data on a set of tightly-coupled machines in well managed data centres to ensure...
that the data will be highly available. Second, it is relatively straightforward to maintain the consistency of replicated data among highly-available machines with low-latency network connections within a data centre. Google’s gfs and Amazon S3 are well-known examples of centralized data storage services.

However, centralized storage also has many drawbacks. First, by storing data using a service offered by commercial companies, user data is subject to the company’s privacy policies. For example, Google’s AdSense program in Gmail reads a user’s emails to display targeted ads. Facebook has already been criticized regarding data privacy issues in its short existence, such as for broadcasting purchase information on News Feed. Companies may also engage in censorship. Normally, this is to eliminate offensive content but there have been some reports of inappropriate censorship. For example, Facebook has been accused by various blogs of censoring their competitors’ names and tampering search results for a presidential candidate. Second, companies only provide sharing services at little or no cost if they can eventually profit from it. This means that specialized applications that are useful to a small group of people may remain undeveloped due to its lack of popularity. It would be beneficial to allow these users to maintain their own storage without paying for expensive data storage services, such as the Amazon S3.

Part 3 of the Report is headed the right to bail. The committee noted that two rights should be recognised in respect of bail in New South Wales. First, in respect of certain minor offences, release on bail by both police and courts should be an absolute right. Second, in all other cases, there should be a presumption in favour of bail – the onus should be on the prosecution to establish grounds for bail refusal. Noting the difficulties for those who seek bail and who are poor, have language problems or find the system unfamiliar, the committee stated, The presumption should be in favour of release on bail rather than limited to the much more illusory right to have bail set. The importance for the presumption of innocence in a liberal democracy could not be more clearly stated. As a matter of social construction it recognises that the target group often includes the poor and those with language problems. At a minimum, they should be seen as positively constructed as deserving and not automatically seen as deviants for whom the politics of punishment has come to dominate much of policy.

There are three requirements for a decentralized data repository supporting multiple writers: durability, availability, and consistency. A storage system is durable if the data can be
Intrinsic Plagiarism Detection System

Chapter 6

Results

recovered in the event of permanent failures, such as disk crashes. Availability refers to how likely a piece of data can be retrieved in the face of temporarily unavailable nodes. The main strategy to achieve high durability and availability is to replicate data onto multiple nodes. In this way, even if a node fails or goes on, the data that was stored on the failed node can be retrieved or recovered from another replica.

Since writers can potentially modify replicas stored at different nodes and many data operations involve many different objects, maintaining consistency among multiple replicas becomes a challenge. The consistency requirements of a system dictate that the same sequence of data modifications are applied to all replicas at different nodes. Consider the example where replica nodes a and b both perform modifications to the repository. If a system does not provide consistency, it is possible that one replica applies writea then writeb whereas another replica applies writeb then writea, resulting in different replica states. If a system is consistent, the final ordering of the operations will be identical for all nodes.

Advances in window systems, availability of high resolution bitmapped displays, and pointing devices such as the mouse make it easy to build sophisticated graphical displays of data. The primary purpose of graphically displaying data is to provide an easy and convenient way to access and manipulate data. Often, people understand better the logical structure of data through visualization than through any other representation. Another advantage of a graphical interface is that it gives immediate feedback of the users' actions on data. For example, when the user adds a new instance, the change is immediately reflected graphically so that the user can see that the instance has been added.

Like their general purpose counterparts, and for many of the same energy related reasons, embedded systems are turning to multicore architectures. Currently, two most prominent synchronization models for shared memory are lock based and speculation based. Our implementation is inspired by a previous work, which proposed a speculation based, hardware technique called speculative lock elision. Thus, this solution achieves lock free execution, in the absence of data conflicts and can be implemented entirely in microarchitecture, without instruction set support or modification to existing cache coherence protocol. In this work, our contribution lies in proposing similar technique in high end embedded systems, keeping energy consumption as an important design
requirement in addition to high performance. Also, our integrated hw sw implementation focuses on hardware simplicity which in turn contributes to energy efficiency.

6.2 Feature Extraction

6.2.1 Document Features

The highlighted words in the above suspicious document are the least frequent words (rare words) which are 24% of the whole document.

Rare words = [resolution, failures, Thus, blogs, structure, alleviated, includes, prosecution, models, becomes, end, offensive, whereas, no, constructed, politics, certain, liberal, sometimes, storing, display, similar, piece, devices, evident, microarchitecture, courts, construction, sophisticated, but, hw, memory, files, was, low, elision, recognised, long, rely, strategy, because, lies, three, within, automatically, ensure, important, most, boards, refers, post, innocence, eliminate, Facebook, does, conservative, Like, deviants, some, Since, often, beneficial, Availability, become, popular, Often, yields, individuals, relied, prominent, emergence, using, performance, collaboratively, possible, absence, consumption, identical, maintaining, adsense, candidate, ads, execution, web, previous, Internet, actions, able, proposed, protocol, centres, concrete, variety, censoring, counterparts, refusal, modification, potentially, any, make, perform, author, mouse, reads, drawbacks, coupled, sle, policies, reliable, these, paying, encyclopedic, general, understand, coherence, offered, whom, due, addition, recognises, expensive, news, reasons, headed, grounds, contribution, etc, called, Centralized, come, police, interface, efficiency, modify, targeted, approach, various, Part, written, importance, illusory, pictures, inappropriate, fails, subject, accused, see, connections, Wikipedia, art, requirement, generally, regarding, Consider, emails, synchronization, criticized, means, achieve, photo, difficulties, failed, state, knowledge, access, find, bulletin, Advances, competitors, face, popularity, sequence, Blogs, centre, Another, added, switching, With, gmail, wanting, concerns, speculative, convenient, Users, latency, relatively, clearly, seek, dictate, Traditionally, were, deserving, Noting, states, matter, instruction, collaborations, target, b, achieves, positively, immediate, Companies, gives, final, 2, durable, ordering, service, little, Flickr, gatherings, temporarily, primary, feed, south, undesirable, turn, messages, could, benefits, conflicts, free, turning, integrated, dominate, Report, publish, cameras, profit, network, event, solution, related, gfs, proposing, entirely, tampering, presidential, transparent, undeveloped, reflected, practice, 3, eventually, main, wales, policy, permanent, window, better, existence, thesis, already, offences, elides, feedback, bitmapped,
representation, allow, so, broadcasting, sw, sets, third, short, purchase, crashes, simplicity, A, how, write, democracy, advantage, information, absolute, −, minimum, locks, organized, specialized, Our, remain, movies, when, challenge, rights, keeping, involve, Normally, cases, known, company, minor, cache, music, multicore, tightly, manipulate, establish, names, Also, modifying, logical, punishment, social, resulting, reports, inspired, engage, search, change, objects, architectures, jointly, commercial, focuses, supporting]

(TF-IDF) Average paragraph rare words belonging to document rare words (in percentage) = 22.69 %
Sentence Length = 22.16
Stop word count = 35.66 %
Word Length = 4.82
Syllable Count = 1.55
Paragraph Length = 6
Noun Count = 17.98 %
Proper Noun Count = 0.52 %
Pronoun Count = 1.80 %
Verb Count = 12.86 %
Adverb Count = 3.23 %
Adjective Count = 7.44 %
Punctuation Count = 9.55 %

6.2.2 Paragraph Features

Paragraph 1
(TF-IDF) Precentage of paragraph rare words belonging to document rare words = 18.57 %
Sentence Length = 14.0
Stop word count = 35.71 %
Word Length = 4.47
Syllable Count = 1.021
Paragraph Length = 5
Noun Count = 15.71 %
Proper Noun Count = 0 %
Pronoun Count = 1.43 %
Verb Count = 14.29 %
Adverb Count = 5.71 %
Adjective Count = 8.57 %
Punctuation Count = 12.86 %

**Paragraph 2**

(TF-IDF) Percentage of paragraph rare words belonging to document rare words = 19.23 %
Sentence Length = 17.33 %
Stop word count = 38.46 %
Word Length = 4.71
Syllable Count = 1.013
Paragraph Length = 6
Noun Count = 18.27 %
Proper Noun Count = 0 %
Pronoun Count = 1.92 %
Verb Count = 8.65 %
Adverb Count = 1.92 %
Adjective Count = 6.73 %
Punctuation Count = 8.65 %

**Paragraph 3**

(TF-IDF) Percentage of paragraph rare words belonging to document rare words = 19.56 %
Sentence Length = 27.0
Stop word count = 37.68 %
Word Length = 4.99
Syllable Count = 1.012
Paragraph Length = 5
Noun Count = 20.29 %
Proper Noun Count = 0 %
Pronoun Count = 1.45 %
Verb Count = 8.69 %
Adverb Count = 1.44 %
Adjective Count = 6.52 %
Punctuation Count = 8.69 %

**Paragraph 4**

(TF-IDF) Percentage of paragraph rare words belonging to document rare words = 14.81 %
Chapter 6

Results

Intrinsic Plagiarism Detection System

Sentence Length = 21.6
Stop word count = 29.63%
Word Length = 4.60
Syllable Count = 1.013
Paragraph Length = 5
Noun Count = 16.66%
Proper Noun Count = 1.85%
Pronoun Count = 0.92%
Verb Count = 12.96%
Adverb Count = 4.63%
Adjective Count = 7.40%
Punctuation Count = 12.03%

**Paragraph 5**

(TF-IDF) Percentage of paragraph rare words belonging to document rare words = 30.5%
Sentence Length = 20.0
Stop word count = 33.5%
Word Length = 4.76
Syllable Count = 2.007
Paragraph Length = 10
Noun Count = 22.0%
Proper Noun Count = 1.0%
Pronoun Count = 3.50%
Verb Count = 15.0%
Adverb Count = 3.0%
Adjective Count = 8.0%
Punctuation Count = 11.5%

**Paragraph 6**

(TF-IDF) Percentage of paragraph rare words belonging to document rare words = 24.27%
Sentence Length = 25.75
Stop word count = 44.17%
Word Length = 4.42
Syllable Count = 1.006
Paragraph Length = 8
Noun Count = 16.01%
Proper Noun Count = 1.45 %
Pronoun Count = 2.42 %
Verb Count = 12.13 %
Adverb Count = 3.88 %
Adjective Count = 5.82 %
Punctuation Count = 7.28 %

**Paragraph 7**

(TF-IDF) Percentage of paragraph rare words belonging to document rare words = 18.18 %
Sentence Length = 22.0
Stop word count = 37.27 %
Word Length = 4.63
Syllable Count = 1.015
Paragraph Length = 5
Noun Count = 14.54 %
Proper Noun Count = 0 %
Pronoun Count = 0 %
Verb Count = 14.54 %
Adverb Count = 2.72 %
Adjective Count = 5.45 %
Punctuation Count = 12.86 %

**Paragraph 8**

(TF-IDF) Percentage of paragraph rare words belonging to document rare words = 17.79 %
Sentence Length = 23.6
Stop word count = 33.10 %
Word Length = 5.21
Syllable Count = 1.014
Paragraph Length = 5
Noun Count = 10.16 %
Proper Noun Count = 0 %
Pronoun Count = 0.84 %
Verb Count = 13.55 %
Adverb Count = 4.23 %
Adjective Count = 9.32 %
Punctuation Count = 7.62 %
Paragraph 9

(TF-IDF) Percentage of paragraph rare words belonging to document rare words = 27.35%
Sentence Length = 23.4
Stop word count = 37.60%
Word Length = 4.99
Syllable Count = 1.014
Paragraph Length = 5
Noun Count = 19.65%
Proper Noun Count = 0%
Pronoun Count = 1.70%
Verb Count = 13.67%
Adverb Count = 3.41%
Adjective Count = 9.40%
Punctuation Count = 7.69%

Paragraph 10

(TF-IDF) Percentage of paragraph rare words belonging to document rare words = 36.70%
Sentence Length = 26.33
Stop word count = 27.21%
Word Length = 5.37
Syllable Count = 2.010
Paragraph Length = 6
Noun Count = 22.15%
Proper Noun Count = 0%
Pronoun Count = 1.89%
Verb Count = 14.55%
Adverb Count = 2.53%
Adjective Count = 8.22%
Punctuation Count = 11.39%

6.2.3 Sentence Features

The sentence features are calculated in the same way the paragraph features are calculated except paragraph length which cannot be calculated for sentences.
6.3 Classification

Using the above features and applying following rules for classification:

**IF** paragraph noun count > (document noun count + 6.71497005988024) **OR**
paragraph noun count < (document noun count - 6.71497005988024) **OR**
paragraph verb count > (document verb count + 5.755688622754491) **OR**
paragraph verb count < (document verb count – 5.755688622754491) **OR**
paragraph word length > (document word length +0.5915568862275449) **OR**
paragraph word length < (document word length – 0.5915568862275449) **OR**
paragraph TF-IDF > (document TF-IDF + 10.68) **OR**
paragraph TF-IDF < (document TF-IDF – 10.68)

**Then** paragraph is plagiarized

**Else** paragraph is not plagiarized

**IF** sentence TF-IDF > (document TF-IDF + 12.48) **OR**
sentence TF-IDF < (document TF-IDF - 12.48)

**Then** sentence is plagiarized

**Else** sentence is not plagiarized

6.4 Plagiarism Report

The plagiarism report is generated which is as follows:
Similarly, the plagiarism report for sentence level detection is as follows:

**54% Plagiarism**

However, this shows an inconsistency where browsing the schema is treated differently from browsing at the data level.

Traditionally, a shared data repository is written and organized by only a single author.

*Plagiarism is commonly remedied by the plagiarism detector in the system.*

For example, if you have a set of beach scenes, you would probably have tans, cars, people, roads, all in common, and if you have a set of beach scenes, there would be oceans, sands, large portion of skies, etc.

Picote allows individuals to upload, organize, and share their own photos.

Harmless has a tremendous capacity to store visual data, but its recall degrades over time.

This trend is evident in the emergence of online collaborations, such as Wikipedia and google groups.

Wikipedia is a popular online encyclopedia that allows users to share their knowledge by modifying the encyclopedia entries.

Google groups is a repository where users can post messages, and share files.

As a concrete example, consider a group of friends wanting to share the photos that they take at gatherings.

At each party, multiple friends take pictures with their own cameras, which yields multiple sets of photos for the same party.

The purpose can also be as Protection of shared images in larger query, to enhance, and it is a threat.

The idea is to keep track of what is added, what is removed, and what is changed.

This idea contributes to the design and development of a decentralized multiple writer data repository that can support a variety of collaborative data sharing applications.
and known as the "blueprint" of the system. Unlike other systems mentioned, it presents data at several different levels.

The system includes a central database and a distributed network of nodes. The central database is used for storing and retrieving data, while the nodes are responsible for processing and disseminating data to other nodes.

The system also includes a peer-to-peer network, where nodes can communicate directly with each other without going through the central database. This provides a more efficient and scalable way to store and retrieve data.

The system is designed to be fault-tolerant, allowing it to continue functioning even if some nodes fail. The nodes are distributed across a wide area, which helps to ensure that the system is available and accessible at all times.

In conclusion, the system presented in this chapter provides a robust and scalable way to store and retrieve data, with features such as fault tolerance and peer-to-peer communication. It is designed to be flexible and adaptable, allowing it to be used in a variety of different environments and applications.
Chapter 7

Conclusion
7 Conclusion

It was a thrilling capability to complete this project. I enjoyed to work on the new technology—Python and its tools and libraries. At this time it is still an evolving and a progressing technology. By the time I was working on the development of this Final Year Project of me, I obviously faced difficulties to understand the growing community of libraries for Python. Because the coding was all manual.

There are many linguistic features which are considered helpful for detecting style changes in a document but for plagiarism detection the best feature so far for our plagiarism corpus is the TF-IDF (Term Frequency –Inverse Document Frequency). Whereas POS tags like noun and verbs are also significant in our model. Word length is also a distinguishing feature for plagiarism detection.

At this time there is no automated tool like Visual Studio by Microsoft or Dreamweaver by Adobe Systems. But these may evolve with time and it will be soon as I see a large number of Software Coders are either shifting to this platform of Python or somehow learning the new aspects of scientific and data analysis.

I passed through all the steps of SDLC, I’ll still say not all but most of the steps were not fully followed as developing a real-world software application is really a difficult process. Still extreme care is taken in consideration while bringing the Intrinsic Plagiarism Detection System to its full bloom.

Besides I learnt so much techniques being a developer, a designer & a software tester:

- How to analyze a system and collect requirements.
- How to design a system.
- How to find and fix bugs and defects.
- How to draw UML diagrams
- How to document the project properly
- How to unit test a module
- How to create integration tests.

7.1 Enhancements that can be done

Following are the enhancements that could be done in the system:

- Integration with external plagiarism detection software.
- Use as a back off tool if external plagiarism detection software fails.
8 Bibliography and References

   http://en.wikipedia.org/wiki/Plagiarism

   http://en.wikipedia.org/wiki/Plagiarism_detection


   http://en.wikipedia.org/wiki/Feature_extraction

   http://en.wikipedia.org/wiki/Binary_classification

   http://en.wikipedia.org/wiki/Accuracy_and_precision

   http://en.wikipedia.org/wiki/Precision_and_recall

   http://en.wikipedia.org/wiki/F1_score