PlagiaBust- A Plagiarism Detection Framework using Text Mining

W.U.C. Costa, K.P.K.C. Jayasinghe, G.G.A.D.N.D. Seneviratne, I.A. Wijesinghe, M. Walpola
Department of Computer Science and Engineering,
University of Moratuwa,
Moratuwa, Sri Lanka.

***Abstract*—With the rapid growth of online resources and data processing tools, today plagiarism is considered as a serious issue in the academic field. Plagiarism can be considered as a theft of intellectual property and many academic institutions are keen to take action against plagiarism. In this paper we discuss about a plagiarism detection framework which enables detecting plagiarism in textual documents. Here we address both the common plagiarism scenarios, namely peer plagiarism detection and Internet plagiarism detection. In our plagiarism detection algorithm, we consider not only copy-paste plagiarism but also paraphrase plagiarism because we consider both methods to be equally important. Also we present the implementation details of our plagiarism detection framework and the core components of our system.**

Keywords-PlagiaBust, Plagiarism, Text Comparison, Paraphrase, Query Creation, Shingle Cloud Algorithm, Preprocessing.

I. INTRODUCTION

The use of electronic documents has become increasingly popular and nowadays almost every academic institute uses online submission of assignments, projects, reports etc. Even though these procedures are followed to gain efficiency it has increased the possibility of someone copying from another's work and presenting it as their own. This kind of action is called plagiarism and academic institutes are keen to detect such actions and penalize those responsible. In a world where securing intellectual property is a challenging task, detecting plagiarism of documents is becoming a very important aspect of the academic environment. But with the advancement of technology and techniques, this task is becoming much more complex. Following are some of the common plagiarism scenarios.

- Copying, modifying or paraphrasing other's ideas and designs without a proper citation procedure.
- Changing the structure of the words or sentences of the original source without explicit reference.
- Giving incorrect citation about the source of a text or a design.
- Directly turning the other person's work as your own work
- Changing the original reference source to a different one.
- Not giving acknowledgment or citation of a unique phrase

Currently there are a lot of software available for plagiarism detection; both commercial and free. Almost all of them have some kind of limitation and limited accuracy of the techniques they use. There are no products that provide fully functional and fully accurate results. In this research we found out that there is no complete fully functional free and open source solution for plagiarism detection. Some of them provide a wide scope for plagiarism testing. Some of this software is capable of checking one document with multiple documents being submitted by the user. Some of them maintain a database with a large volume of papers, books, magazines, and blogs and compare them to provide plagiarism reports. In addition, they test with cached and live Internet pages. In these tools, plagiarism is extracted through a large number of testing phrases.

Mainly there are two types of plagiarism that can be identified within the academic context; namely peer plagiarism detection and Internet plagiarism detection. Peer plagiarism is the case where a student copies another student's work implicitly. Internet Plagiarism is where a student copy online documents and present it as their own work. With the rapid growth of Internet resources, possibility of Internet plagiarism is becoming higher than ever before.

II. RELATED WORK

Currently there are a few commercial and non-commercial products available for plagiarism detection. Those products have been developed based on different plagiarism detection technologies.

A. Turnitin

Turnitin is a commercial web based product and users can subscribe via a web portal. The user uploads the documents that need to be checked for plagiarism. Turnitin then checks for plagiarism in those documents as well as word phrases that have been copied directly from the Internet [1]. Turnitin also has the ability to check for documents in the paper mills. Technical details were not revealed since this is a commercial application.

B. CopyCatch

CopyCatch is a standalone application and does not require web access. Therefore, it does not check for internet plagiarism. It checks for plagiarism among multiple submitted documents. It can be installed on a network for the use of multiple users. The CopyCatch system works based on the lexical similarity between two texts. It removes the irrelevant texts using lexical cohesion techniques and then it will strictly check for hapaxlegomena words. Hapaxlegomena words are the words that only appear once in a document. They believe that similar hapaxlegomena words between two documents indicate a possible plagiarism activity [2].

C. Essay Verification Engine – EVE2

EVE2 is basically an interface for searching through the Internet for plagiarism activities. It only checks for documents copied from the Internet and can't compare documents with each other. EVE2 is a commercial application, hence its technical details are not revealed.

.

D. WordCheck

WordCheck is another standalone application that does not check for Internet plagiarism. It only detects plagiarism among a set of documents. From a technical aspect, the system uses a simple method to detect plagiarism. It counts the number of occurrences of each word in every document and compares the results.[2]

E. JPlag

JPlag is a web based service and all the files that need to be examined must be in a single folder. JPlag considers all of them as belonging to a single submission. JPlag uses a "Greedy String Tiling" approach which is discussed later in this report. [3]

F. Plagiarism.Org

The tutor is facilitated to register his/her class in the system and students are allowed to upload their assignments. At the end of the process, a report is e-mailed to the tutor. Implementation details of the system are not disclosed, since it is a commercial application.

III. DESIGN AND IMPLEMENTATION

A. System Overview

The Core of our application consists of 5 main modules namely Data Extraction Module, Data Preprocessing Module, Document Similarity Detection Module, External Source Detection Module and Report Generation Module. Data Extraction Module captures the files submitted by the user to detect plagiarism and extract the data on each file by converting them into text format. Data Preprocessing Module is responsible for processing the output documents of the Data Extraction Module using approaches such as synonym replacement, stemming and stop word remover etc. The details of each of these methods will be discussed later in this paper. The Document Similarity Detection Module is responsible for accessing the document content, compare and return the possible plagiarized phrases using an advanced plagiarism detection algorithm. Detecting the possible plagiarized Internet sources and PlagiaBust Server Sources is done by the External Source Detection Module. The details of the possible plagiarized document with detailed analysis will be handled by the Report Generation Module. Each of these Modules with their detailed description and the functionalities will be discussed later in this paper.

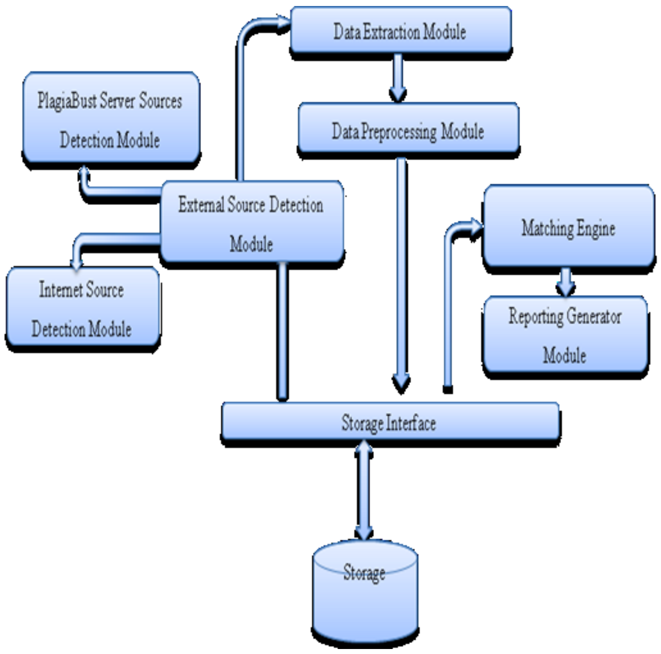
Data Extraction Module
PlagiaBust Server Sources Detection Module
Data Preprocessing Module
External Source Detection Module
Matching Engine
Internet Source Detection Module
Reporting Generator Module
Storage Interface
Storage

Figure 1. System Overview

B. Data Extraction Module

There are various document formats that are available nowadays in student work. Most commonly used document formats are Microsoft office "docx", "doc" formats, "PDF" format, "RTF" format and plain text format. No matter in which format the documents are supplied to the system, it should be able to convert them in to plain text format. So the initial preprocessing stage requires extracting text from these types of documents. This module outputs text from different file formats and gives this text as input to the pre-processor module. This procedure enhances the document processing speed.

C. Data Preprocessing Module

Main objective of the pre-processing module is to generate a normalized form of each document. It means we have to take each submitted document and make them to a single document format. Usually text documents contain lots of unwanted words which are not relevant in document comparison in the comparison phase of plagiarism detection. Therefore, a stop word list [4] is used to filter out the most common words in English language that are not carrying information about the content of the document.

In the next step, we stemmed all the words using Porter stemming algorithm [5] to get the stemmed form of words, so that the tense and the form of the words become uniform in all the inputted documents. Then, we use synonym replacement. The motivation for using synonym recognition comes from considering human behavior, where people may seek to hide plagiarism by replacing words with appropriate synonyms. If a sufficient number of words are replaced by synonyms, then most of the common copy detection methods fail. So, the best solution is to transform words having the same meaning into a unique identifier. A consideration has to be given to words that have more than one meaning; if a significant impact on the accuracy is expected, a disambiguation process is required to determine the appropriate meaning.

Synonym Replacement is another key task in text preprocessing because one of the common ways to change the original sentence is to replace some words with their synonyms. Considering this factor, paraphrase detection and text comparison methods can benefit from use of synonym replacement strategy. WordNet [6] is the most well known lexical database for the English language. As we described in our literature review, WordNet provides facilities to find synonyms for a given word. A set of synonyms for a given word is called a "synset". There are so many Java APIs for accessing the database like JWNL, JWI, lucene – wordnet etc. From those APIs we have selected lucene-wordnet API in our project. Compared to other APIs lucene-wordnet uses a memory map and accessing time for a synset is much less than the others.

D. External Source Detection Module

External source detection module is responsible for downloading possible sources of plagiarism from the Internet. At the most basic level, the system can only compare two documents and provide a measure on whether plagiarism is present in that work. So to detect places where it has been copied from Internet, first the system must download possible sources from the Internet. Internet has billions of documents, so it is not easy to predefine the sources where a student may use to copy. Obvious solution is to dynamically search for documents from the Internet where similar texts are presented.

This module uses Internet search services (optionally Bing or Google) to find sources for some document. By providing a set of word sequences generated from the suspicious document and looking for the results retrieved by the Internet search service it is possible to detect sources which have similar text to the suspicious document. Even though the query made by module is very specific, there still can be millions of results for a single query. Internet search services have a number of measures when it comes to sorting out these results.

Query selection algorithm is the key bottleneck of performance for Internet source detection. A list of queries selected to represent the document is the key point to identifying download sources from Internet. If this list is unnecessarily large then the Internet search time will increase, which consequently increases the overall system response time. Moreover, if the query list is too large then we have to be more concerned about constraints set by the Internet search service. And if the query list is large then there will be a huge number of collective hits, where the system has to sort it and find the web links with the highest frequency of occurrence. As the results suggest, if the document is nearly copied from internet sources, selecting a huge list of queries will not improve the result a great deal.

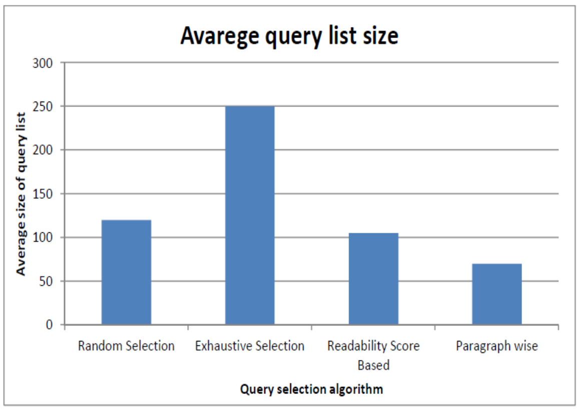

Figure 2. Avarege query list size

According to the tests we have carried out which is presented in the above graph showing the average number of queries generated by each query selection algorithm, exhaustive query selection algorithm recorded the maximum number of queries and the paragraph wise algorithm the minimum. As a matter of fact the amount of time taken for retrieving the possible sources for a particular document will depend on the time spent on querying the internet live search service. Furthermore, if more queries are used, then the list of all retrieved internet links will be very large which need more time to be sorted and find out the links with the maximum number of occurrences. So an algorithm with a minimum will be more preferable. But it is important that the result should be accurate to a great extent. Once these sources are marked, they are downloaded to a local directory and checked thoroughly against the suspicious document. The document content can be from different sources. In the Internet there are websites with same content. For example Wikipedia articles can be found in other websites as well. Retrieving these both sources can increase the precision of the algorithm but it will not make much of contribution to finding all the sources that the suspicious document copied from. So recall can be considered as the most significant measure of performance of the query selection algorithm.

The paragraph wise query selection algorithm selects the least set of queries and it has recorded a greater level of accuracy. This is a very important design decision as the system should not unnecessarily compromise its performance. Readability score based algorithms have greater level of accuracy irrespective of the level of the copied content. But it has shown poor values for recall. Also creating a query list from the readability based algorithm involves complex calculations compared to the other three algorithms.

Considering all the facts we can say that paragraph wise query selection algorithm is the most appropriate solution. It has produced less overhead to the system by generating the least set of query lists with fairly good accuracy. With huge query spaces, random and exhaustive selection algorithms are infeasible to use, and readability score based algorithms have a poor level of recall.

Internet source detection is one of the most time consuming tasks of overall plagiarism detection process. It depends on lots of circumstances like internet connectivity and size of possible sources that need to be downloaded to the system. Querying the internet live search service can take a huge amount of time depending on the size of the query list. If we are going to check for hundreds of documents this time gets multiplied. The other issue with using Internet live search service is that only a limited number of maximum queries can be made. If the collective number of queries to be made overran this number, then the system will not be able to detect Internet sources for some documents. If the size of the document corpus is very large these issues make Internet source detection using Internet live search services, an infeasible one.

PlagiaBust web search service is introduced as the alternative to Internet search service. This is a very practical solution if the PlagiaBust software is used by a University or any other education institute. Then university can maintain a PlagiaBust search server so that all client programs from university can access to store and query. PlagiaBust web search service will index all the required documents and provide search service for PlagiaBust client programs. By this method, the system need not consider about overrunning of query space. Furthermore if the search service is deployed in a Local area network, then querying the search service will be faster in large scale compared to Bing Internet live search service. More importantly downloading sources will be faster by few times.

E. Document Similarity Detection Module

1) Text Comparison

One of the most important aspects of this research is to find a suitable textual comparison algorithm which enhances the accuracy of plagiarism detection. The selected text comparison algorithm should be able to accurately measure the similarity between two given texts and indicate whether there is a possibility of plagiarism. There are several text comparison algorithms such as Cosine Similarity, Euclidean Distance, Greedy String Tiling [7] and Shingle Cloud [8] that can be used for this purpose. We conducted our research on these algorithms and tested them on various inputs.

Following graph shows the average similarity measures given by the algorithms we considered for nearly 60% copied set of texts.

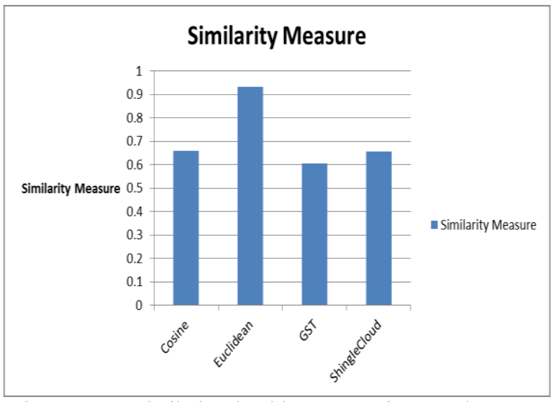

Figure 3. Text similarity algorithms comparison graph

According to the above graph we can see that similarity measures of Cosine, GST and ShingleCloud algorithms are in similar range while the similarity measure given by Euclidean distance algorithm is highly deviated from those values. As a result we ignored Euclidean distance as it gives values much larger than we anticipated. So we had to select one from Cosine, GST and Shingle Cloud algorithms but Cosine Similarity algorithm does not provide a set of matching phrases among two texts so it could not be used. GST algorithm is based on a greedy approach and sometimes it can ignore smaller matching phrases and look for single larger matching phrase. Taking these constraints in to consideration Shingle Cloud was selected as the text comparison algorithm to be used.

“ShingleCloud” is a string comparison algorithm developed using the n-gram-overlap approach [8]. The two strings that are to be compared are called as the “needle” and “haystack”. The comparatively shorter string is called the needle and the other one is the haystack. Before proceeding further the two strings are preprocessed using techniques such as “stop word removal”, “stemming”, etc. The next step is to extract n-grams from the haystack using a sliding window of size “n”, which are ultimately called shingles. Then those extracted shingles are stored in a data structure suitable for fast lookups. Then a bit string is created according to the n-gram considered. If there is a corresponding match for a particular n-gram in the needle, 1 is appended or otherwise 0 is appended to the bit string. At the end, a bit string is generated corresponding to the haystack and needle containment. This bit string is used to calculate the containment measure to compare the two strings. There are two important parameters for declaring a match. The first one is the minimal number of consecutive ones, which is the minimum number of consecutive ones that need to occur to consider it as a match. The second one is the maximal number of zeros, which is the maximum number of zeros that can be included in a single match.

2) *Selecting Parameters for ShingleCloud*

There are two important parameters for ShingleCloud algorithm. One is the N-Gram size and the other one is the minimum number of ones in a single match.

a) *Selecting N-Gram size*

ShingleCloud algorithm divides the given text into N-Grams and these grams are used for comparison. Selecting a suitable value for the size of a gram is important since it directly affects the similarity value. Following table shows the change in similarity value with the change in N-Grams.

TABLE I. CHANGE IN SIMILARITY WITH N-GRAM SIZE

Size of N-Gram	**No of Matching Shingles**	**Containment in Haystack**	**Containment in Needle**	**Similarity**
1	4	0.7878788	0.82539684	0.806638
2	4	0.76515156	0.8015873	0.783369
3	5	0.76515156	0.8015873	0.783369
4	5	0.76515156	0.8015873	0.783369
5	4	0.7348485	0.76984125	0.752345
6	4	0.7348485	0.76984125	0.752345

We can see that with the increase of n-gram size similarity value decreases. If the n-gram size is smaller, the algorithm looks for similar tokens with shorter length. If the n-gram size is larger it looks for longer matching phrases. To have a balance between these two extremes we selected n-gram size to be 4.

b) *Selecting minimum number of ones*

Minimum number of ones in a single match is the number of ones a match should have to consider it to be a match.

TABLE II. SIMILARITY WITH MINIMAL NUMBER OF ONES

Minimal Number of Consecutive Ones	**No of Matching Shingles**	**Containment in Haystack**	**Containment in Needle**	**Similarity**
1	8	0.6136478	0.637899	0.62577
2	8	0.6136364	0.6377953	0.625716
3	8	0.6136364	0.6377953	0.625716
4	7	0.5681818	0.5905512	0.579367
5	6	0.5151515	0.5354331	0.525292
6	4	0.39393938	0.4094488	0.401694

Looking into these results we can see that with the increase of minimum number of ones similarity decreases. If the selected value is too small similar n-grams will be matched and if the value is too large it can easily miss shorter matching phrases. So to have a balance between these two, minimum numbers of ones was selected as 4.

IV. ParaPhrase Detection

Some of the plagiarism cannot be detected using normal text comparison as paraphrasing is one of the techniques done in plagiarism. Two sentences are paraphrased if they "mean the same thing". If we think of a more broad definition to paraphrasing we can say paraphrase methods recognize, generate, or extract paraphrases, meaning phrases, sentences, or longer texts that convey the same, or almost the same information [9]. For an example consider following three sentences:

(1) Wonderworks Ltd. constructed the new bridge.
(2) The new bridge was constructed by Wonderworks Ltd.
(3) Wonderworks Ltd. is the constructor of the new bridge.

We can say that 1 and 2 are paraphrased clearly. But we cannot say that 3 were paraphrased from1 and 2 directly

In our method we have used the Semantic Similarity calculation method to detect paraphrased sentences.

The basic function to semantic similarity value:

$$sim(T_1,T_2) = \frac{1}{2}\left(\frac{\sum_{w\,\epsilon\,\{T_1\}}(maxSim(w,T_2) * idf(w))}{\sum_{w\,\epsilon\,\{T_1\}} idf(w)} + \frac{\sum_{w\,\epsilon\,\{T_2\}}(maxSim(w,T_1) * idf(w))}{\sum_{w\,\epsilon\,\{T_2\}} idf(w)}\right)$$

In our algorithm we have to add some other similarity calculations like synonym similarity and sentence length similarity. For each sentence pair we calculate the similarity value using similarity calculation function. We modified the similarity calculation function as below.

$$\text{similarity }(T_1,T_2) = \text{sim }(T_1,T_2) + \text{lengthSimilarity }(T_1,T_2)$$

Here,

$\text{lengthSimilarity }(T_1,T_2) = \text{score for lenth differences between two sentenses}$

In the sim(T_1,T_2) function maxSim(T_1,T_2) also changed:

$$maxSim(w,T) = maxSim(w,T) + synonymSimilarity(w,T)$$

Here,

$synonymSimilarity(w,T) = \text{number of synonyms of the word w in sentense T}$

with the use of similarity measure for two sentences, we decide whether two sentences are paraphrased or not. If the value is greater than the threshold value we take those sentences as paraphrased and if the value is less than threshold value (we have selected the threshold of 0.27) we take those sentences as not paraphrased. Next, the paraphrased sentences are sent to the reporting module.

V. PARAPHRASE DETECTION METHOD ACCURACY MEASURES

For testing the accuracy of the method, we have used the Microsoft Research Paraphrase Corpus [10]. To compare with other methods we have used the common accuracy measures (Accuracy, Precision, Recall, and F-measure).

TABLE III. PARAPHRASE DETECTION METHOD ACCURACY VALUES FOR THRESHOLD VALUES

Threshold	**Accuracy**	**Precision**	**Recall**	**F-measure**
0.25	71.01	73.37	89.25	80.54
0.26	71.22	73.87	88.50	80.53
0.27	71.31	74.38	87.43	80.38
0.28	71.03	74.64	86.20	80.00
0.29	70.84	75.09	84.74	79.62
0.30	70.47	75.37	83.28	79.13
0.35	69.14	77.00	76.00	77.00

In this section we compare our paraphrase detection method with other available methods in the literature. Microsoft Research Paraphrase Corpus was used to compare because other available methods have results for this tool as well. All the accuracy measures used in above analysis were used in this comparison as well. Accuracy, Precision, Recall, and F-measure values for other methods were taken from published research papers and a survey paper [9].

TABLE IV. COMPARISON OF AVAILABLE METHODS

Method	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F-measure (%)**
Corley & Mihalcea [11]	71.5	72.3	92.5	81.2
Das & Smith[12]	76.1	79.6	86.1	82.9
Finch et al. [13]	75.0	76.6	89.8	82.7
Malakasiotis[14]	76.2	79.4	86.8	82.9
Qiu et al. [15]	72.0	72.5	93.4	81.6
Wan et al. [16]	75.6	77.0	90.0	83.0
Zhang & Patrick [17]	71.9	74.3	88.2	80.7
Our Method	**71.31**	**74.38**	**87.43**	**80.38**
BASE1	66.5	66.5	100.0	79.9
BASE2	69.0	72.4	86.3	78.8

In above comparison we have mentioned two base cases. BASE1 classifies all pairs as paraphrases. That means the recall value is 100% this is because there are no false negative values. BASE2 classifies two sentences as paraphrases when their surface word edit distance is below a threshold value.

Even though our method has lower accuracy value than other methods, all the other measures are quite similar to other methods. Our method is a simple method which uses semantic similarity measure with synonym similarity and length similarity. So the calculation is quite simple compared to other available methods. As other methods use complex calculations and syntactic analysis to detect paraphrases, they take some time to detect paraphrases from documents. Therefore for an application like plagiarism detection our method is more suited.

VI. REPORT GENERATION MODULE

Report Generating module handles the overall presentation of the plagiarism-check results between documents. It enables the user to visualize the plagiarism results in a graphical manner so that the user can easily evaluate the plagiarism between documents. It mainly consists of key components such as text highlighting component, connectivity graph generating component, final report generating component etc. which provide visualization of the plagiarism results.

- Text highlighting component- This component provides text highlighting capability for onscreen viewing of two suspected document. It highlights the possible plagiarized phrases between two documents in the onscreen view. The color of the highlighter changes depending on the matching phrase.

- Connectivity graph component: The plagiarism results are shown as a connectivity graph inside the module. Here the nodes represent the document and the edges between them represent that the two documents are plagiarized. The colors of the edges change depending on the plagiarized percentage between documents.

- Final report generating component: The plagiarism check results are printed in a report which can be exported into many common document formats such as PDF, HTML, DOC, DOCX etc. This report can be seen inside the module also using a viewer. It has zooming capability and the paging capability also.

- Onscreen view component- The suspected document is displayed inside the module along with the plagiarized phrases highlighted in red color. When the user clicks on the phrases it displays the suspected source of the phrase. If it's an internet source it will display the content of the web page in the integrated browser component.

VII. RESULTS AND DISCUSSION

For the performance analysis of PlagiaBust we selected a plagiarism detection corpus downloaded from [18], which contains sample documents. From that corpus we created a dataset suitable for our context. Following is the constitution of that dataset.

TABLE V. PLAGIARISM DETECTION CORPUS CONSTITUTION

No of Documents	Original Documents	Near Copy	Light Revision	Heavy Revision	No Plagiarism
76	5	14	15	16	31

- Original Documents – These documents are answers to five questions given and they are not plagiarized.
- Near Copy – The document is created from copying text from an original document.
- Light Revision – The document is constructed based on the content of an original document it may contain some copy pasted parts. Some of the copied contents are altered in some basic ways including substituting words and phrases with synonyms and altering the grammatical structure. But the order of information found in the sentences is not changed.
- Heavy Revision – The document is constructed based on the content of an original document but the texts are rephrased to generate a document with having a same meaning as original text. Some of the sentences are split into more sentences and no restrictions are given on how the text should be altered.
- No Plagiarism – The document answers the given five questions not based on the original documents. No part of the document is plagiarized from other documents or the Internet.

TABLE VI. PLAGIARISM DETECTION RESULTS

	Near Copy	Light Revision	Heavy Revision	No Plagiarism
Sample	14	15	16	31
Detected	10	7	6	0

Accuracy = 1 = 1.0000

$$\text{Recall} = \frac{TP}{TP+FN} = \frac{10}{10+4} = 0.7142$$

Looking about the results we can see that there are no false positives given by the system, and it has a good accuracy of 0.9111. In this calculation we didn't consider lightly and heavily revised documents because there is no indication to actually know whether they have been plagiarized or not. But we can see that certain documents from those two categories are still detected as plagiarized documents by the system.

The overall performance of the PlagiaBust was measured by the time spent for the system to produce results. This period covers the time taken from the start at preprocessing documents to finish at document comparison. The experiment was carried out using several test cases while varying the no of files processed and the size of a single file. Measured results are shown below.

TABLE VII. PLAGIABUST PERFORMANCE WITH FILE SIZE AND NO OF FILES

No of files	Size of a single file(KB)			
	2	4	8	16
10	2.959	3.567	6.159	11.298
25	8.938	9.497	13.962	31.355
50	15.081	20.532	54.378	182.741
75	22.432	49.002	148.294	596.478
100	31.627	106.725	323.269	945.921

Through observing above graphs we can say that when numbers of files are increased, execution time has also increased with that. This is quite obvious since when there are more files to process it takes more time to preprocess, index and compare them. We can also observe that with the increase of file size, execution time also has increased. The primary reason for this is with larger files, system has to process larger data so overheads in using memory and processing power causes PlagiaBust to slow down its operations.

VIII. FUTURE WORK

In this paper we have presented a method for the comparison of documents aimed at spotting plagiarism only in unstructured English text. For future improvements we can extend this work to detect plagiarism in code assignments and also in other languages. Paraphrase detection can be improved to get more accuracy. Currently the described method doesn't have the support for getting the evaluator's feedback on the plagiarism results between documents. This can be further improved to get the evaluator's feedback on the results obtained and perform a new check according to the feedback on the plagiarized phrases. We can also extend this plagiarism detection method to support student profile system, where each student in a university has a dedicated profile including the past assignments submitted and statistics about the writing style etc. This can be implemented in the future to enhance the plagiarism results accuracy of the student assignments.

REFERENCES

[1] (2010, December) turnitin. [Online]. http://turnitin.com/static/index.php"

[2] Paul Clough, "Plagiarism in natural and programming languages: an overview of current ," Department of Computer Science, University of Sheffield, 2000.

[3] Guido Malpohl, Prechelt Lutz, and Michael Phlippsen, "JPlag: Finding plagiarisms among a set of programs," Fakult ¨ at f ¨ur Informatik Universit ¨ at Karlsruh, D-76128 Karlsruhe, Germany,.

[4] (2011, March) Stop Word List. [Online]. http://www.lextek.com/manuals/onix/stopwords1.html

[5] M.F. Porter, An algorithm for suffix stripping., 1980, vol. 14, ch. no. 3, pp. 130-137

[6] WordNet. (January, 2011) A lexical database for English.[Online]. http://wordnet.princeton.edu/wordnet/

[7] Michael J Wise, "Running Karp-Rabin Matching ," The University of Sydney, ISBN 0 86758 669 9 , March 1993.

[8] Arno Mittelbach, James Cummings, Christoph Rensing, Ralf Steinmetz Lasse Lehmann, "Automatic Detection and Visualisation of Overlap for Tracking of Information Flow," KOM Multimedia Communications Lab, Technische Universität Darmstadt, Rundeturmstr. 10, 64283 Darmstadt, Germany and Research Technologies Service, Oxford University Computing Services - University of Oxford, 13 Banbury Road, Oxford, OX2 8NP, UK ,.

[9] Prodromos Malakasiotis Ion Androutsopoulos, "A Survey of Paraphrasing and Textual Entailment Methods," *Journal of Artificial Intelligence Research*, vol. 38, pp. 135-187, May 2010.

[10] (2011, May) Micosoft Research. [Online]. http://research.microsoft.com/en-us/downloads/607d14d9-20cd-47e3-85bc-a2f65cd28042/

[11] C. Mihalcea and R. Corley, "Measuring the semantic similarity of texts," in *Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, 2005, pp. 13–18.

[12] D. Smith and N.A. Das, "Paraphrase identification as probabilistic quasi-synchronous recognition," in *47th Annual Meeting of ACL and the 4th Int. Joint Conf. on Nat. Lang.*, Singapore, 2009, pp. 468–476.

[13] A. Hwang, Y. S. Sumita, and E. Finch, "Using machine translation evaluation techniques to," in *3rd Int. Workshop on Paraphrasing*, Jeju Island, Korea, 2005, pp. 17-24.

[14] Y. Malakasiotis, "Paraphrase recognition using machine learning to combine similarity measures," in *47th Annual Meeting of ACL and the 4th Int. Joint Conf. on Nat. Lang.*, Singapore, 2009.

[15] L. Kan, M.Y. Chua, and T. Qiu, "Paraphrase recognition via dissimilarity significance classification," in *Conf. on EMNLP*, Sydney, Australia, 2006, pp. 18-26.

[16] R. Neumann and G. Wang, "An divide-and-conquer strategy for recognizing textual entailment," in *Text Analysis Conference*, Gaithersburg, 2008.

[17] Y. Patrick and J. Zhang, "Paraphrase identification by text canonicalization," in *Australasian Language Technology Workshop*, Sydney, Australia, 2005, pp. 160-166.

[18] Clough P. and Stevenson, Developing a Corpus of Plagiarised Short Answers, Language Resources and Evaluation: Special Issue on Plagiarism and Authorship Analysis, In Press. [Download], 2011 September.