Duplicity Detection System for Digital Documents

Ranjeet Singh, Chiranjit Dutta

Abstract- Plagiarism detection is a challenging problem. Today thousands of documents are present on the net but there are no proper tools to guarantee their uniqueness in such a great domain. PDF documents form a significant portion of this vast database. Copy detection in digital document database may provide necessary guarantees for publishers and newsfeed services to offer their valuable work for others perusal. We consider the case of comparing a Query Document with a Registered Document .Plagiarism detection techniques are applied by making a distinction between natural and programming language. In this paper we have implemented SCAM (standard Copy Analysis Mechanism) which is relative measure to detecting copies based on comparing the words and lines frequency occurrences of the new document against those of registered documents. These tests involve comparisons of various articles and show that in general this scheme performs pretty well in detecting documents that have Exact, Partial and Trivial overlap.

Keywords: Plagiarism, SCAM, WordNet, Registered Document, Query Document

I. INTRODUCTION

A document may be seen as a list of terms, list of key-words, and set of related words or concepts. It seems obvious that it is not the same to have just simple words as the basic items of the representation, that to have a set of words (may be structured someway) with a representation of their meaning [1]. The analysis on documents' contents can be semantic or statistical. Regarding the document representation, a commonly used Information Retrieval approach is the adoption of the Vector Space Model, where the similarity score between two documents is calculated using the cosine formula, resulting the cosine of the angle

of the angle between the two corresponding vectors. This representation is also called a "bag-of-words", since the list of word positions are not maintained, hence relation-ships between words are missed [2]. This seems not to be appropriate for a plagiarism detection system that works on text documents, also cosine formula has issues when documents differ in size. In natural language, a sentence may be seen as the fundamental part of a discourse and the minimum unit to express a concept.We considered a good norm to build a sentence-level system to compare documents using semantic analysis. Sentence boundaries allow us to keep track of meaning and contest of terms, maintaining information about their position and mutual relationships.

A phrase is extracted from each document every time a particular punctuation mark is met; the vocabulary of terms is expanded with synonyms through Wordnet, trying to cover paraphrasing. To search for source containing suspicious phrases a search engine is required. Usually plagiarism detection involves Internet sources and web search engines which is free, easy and fast way of detecting plagiarism. The user can copy and paste or type in suspicious phrases taken from suspected plagiarized work into a search engine in an attempt to find on-line material containing the suspicious phrases" [3]. Unfortunately they are not open source and working with them means that there is no possibility to tweak the code according to your requirements and consequently, user lacks complete control about elaboration and results. In order to avoid such limitations we decided to use our own search engine based on Swing java library. This is a major benefit of using an open source search engine, since one can tweak the calculation of the score for a document to the required specifications. The scoring and similarity calculations are transparent and one can build similarity classes that are appropriate for required domain. For each sentence in each document, several searches are launched, trying to cover all the possible forms of a plagiarized phrase. The similarity matches are obtained with (using) SCAM algorithm and they are displayed on a GUI. The idea is to present a list of plagiarized documents ordered by similarity score. Thus the user has the possibility to visualize in detail the compared parts. To evaluate the effectiveness of the detection system, "EXACT, PARTIAL, TRIVIAL, or NO Match with the Registered Document, and then further based on it, we can decide whether to add Query Document into the database or not" are implemented.

II. II. LITERATURE SURVEY

A. Copy Detection Preliminaries

In this section, we present the architecture of a generic copy detection server and introduce relevant.

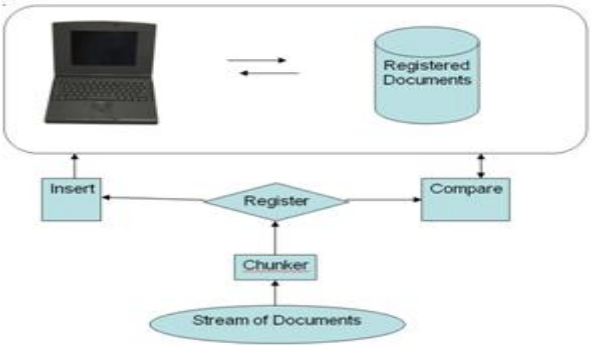

Figure 1: A Generic Copy Detection Server

Terminology. We also give a brief summary of some issues that need to be considered while building a copy detection server such as data structures, and the textual units used for comparison.

Manuscript Received on November, 2012.

Mr. Ranjeet Singh, Faculty of information Technology, SRM University, NCR campus, India.

Mr. Chiranjit Dutta, Faculty of information Technology, SRM University, NCR campus, India.

B. Copy Detection Server Architecture

In Figure 1, we see the architecture of a generic copy detection server with a repository of registered documents. (The repository is shown to be centralized, but in practice may be distributed.) We define chunking of a document to be the pro cess of breaking up a document into more primitive units such as sentences, words or overlapping sentences. Documents that are to be registered are chunked and inserted into the repository. New do- cuments that arrive are chunked into the same units and are compared against the pre-registered do- cuments for overlap. In subsequent sections we will consider different chunking units and document similarity measures.

Let W represent the vocabulary of the chunks, that is the set of occurrences of all distinct chunks in the registered documents. Let wi refer to the ith chunk in the vocabulary. Let the size of the vocabulary (number of distinct chunks) be N.

C. Inverted Index Storage

We propose using an inverted index structure (as in traditional IR systems []) for storing chunks of the registered documents. An index of the chunks in the vocabulary is constructed and maintained at registration time.

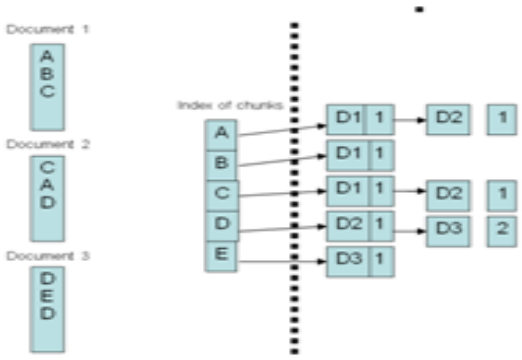

Figure 2: Inverted Index Storage Mechanisms

Each entry for a chunk points to a set of postings that indicate the documents where the chunk occurs. Every posting for a given chunk wi has two attributes (docnum, frequency), where docnum is a unique identifier of a registered document, and frequency is the number of occurrences of wi in document with ID docnum. In Figure 2, we illustrate the index structure with three registered documents. The letters \a" through \e", which represent the chunks in the documents, constitute the vocabulary with N=5. For instance, chunk \d" has two postings representing that it occurs once in document D2 and twice in documents D3.

When a document D is to be compared against the pre-registered documents, the chunks of Dare looked up in the registered documents index.

This means that only the documents that overlap at the chunk level will be considered using this index mechanism. Hence the total number of looks ups on the index is the number of distinct chunks that occur in the document D.

D. Units of Chunking

As defined earlier, chunking involves breaking up a document into more primitive units such as paragraphs, sentences, words or overlapping sentences. The unit of chunking chosen for copy detection is critical since it shapes the subsequent overlap search cost and storage costs as outlined below.

- **Similarity level**: The bigger the chunking unit the lower the probability of matching unrelated documents. For instance, two unrelated documents may both have a sentence like "This research was funded by NSF" as part of a paragraph. If the chunking unit is a paragraph, the two documents will probably not be detected as an overlap, while they will be detected if the chunking unit is a sentence. On the other hand, the bigger the chunking unit, the higher the probability of missing actual overlaps. For instance, consider two paragraphs that share 5 out of 6 identical sentences. With paragraph chunking, no match will be detected, while with sentence chunking 5 out of the possible 6 units will be detected as matching.
- **Search Cost**: The larger the chunk, the higher the potential number of distinct chunks that will be stored. For instance, as the collection of documents grows, we expect the number of distinct sentences that will be stored to be higher than the number of distinct words. This is because beyond a certain point the number of new words introduced into the vocabulary will be low as opposed to the near-linear growth of sentences/ paragraphs. Hence we see that

the potential size of the chunk index is higher when the chunking unit is chosen is larger. Of course the number of postings per chunk is larger when the chunking unit is small (as in words).

However, we see one advantage for small chunking units. A small chunking unit increases locality. That is most documents will have a relatively small working set of words rather than sentences. Consider the frequency distribution of N words to follow Zipf's Law [12,15, 11]. If the words are ranked in non- increasing order of frequencies, then the probability that a word w of rank r occurs is

$$P(w) = \frac{1}{r * \sum_{v=1}^{N} 1/v}$$

If we assume a vocabulary of about 1.8 million words [17], about 40,000 (about 2% of 1.8 million) words constitute nearly 75% of the actual occurrences of words thereby increasing the effects of caching.

That is, by retaining the most popular words and their associated postings in main memory, we may be able to reduce the number of accesses to the disk resident portion of the index. With sentence chunking, we expect the access pattern to be more random due to very large size of the sentence vocabulary.

As we have argued, word chunking may lead to more locality during comparisons. In addition, word chunking has the potential to detect finer (e.g. ,partial sentence) overlap, which may be especially important with in formal documents that may not have a clear sentence structure. As discussed earlier, COPS sometimes has problems detecting sentence boundaries.

However, before we can use word chunking, we need to determine a good scheme for comparing documents. Recall that for sentence chunking, comparison was straightforward: if X of the Y sentences in document D1 appear in D2 then the overlap is X*100/Y [6].

Unfortunately, this simple scheme breaks down for words: the fact that D2 has many of the words of D1 does not necessarily mean they do not necessarily mean they overlap. In the next section, we propose scheme based on relative frequency of words that we have empirically found to be effective.

III. IMPLEMENTATION

In this section we present an implementation of our plagiarism detection system based on Query document (Line and sentence matching). The main tools and utilities used to develop our application are vector space and relative frequency model. However, details of the document structure and the main classes of code carrying out the execution also described in this section. The programming language chosen is java, the motivation is given by the fact that the native version of partial match is written in this language and this can easily help us to manage and possibly modify it.

A. Detection Steps

In this section we report a detailed overview on the main methods implemented and steps followed to achieve a detection system. There are three main steps:

- Processing the registered document: The registered document is given input is processed in order to tokenize it in a list of words; stemming and stop word removals are applied.
- Searching on the index: The index is questioned in order to retrieve a match between the registered document and query document.
- Evaluating Similarity: Using SCAM formula, a similarity measure is calculated between registered document and the query documents.

The code is divided in three Java packages:

- Core: Handles the project execution; all the Java classes implementing the three modules are included in this packages are: which are: javaprogram.java, checkplagiarism.java, dupdetector.java.
- Utils: Includes some Java Classes for example, Config.java: handles information about the configuration and relative paths contained in the file Config.dat, PDFTextParser.java: contained methods to support extraction of tokens from raw text convert into PDF format.
- GUI: contains graphical user interface.

B. Processing the registered document

In the step input data is processed and it is called a registered document. The class in charge of performing this task is ProcessTest. The main method in this class is extract token; it has two main objectives: fill a data structure (a Java HashMap) and fill the table testphrase. The method uses the same Analyzer which is used to Query document; this is very important in order to keep consistency in the way the text is processed and in order to achieve better results. The registered document is splited in the same ways query document have been splited by using regular expressions and the same logic is used to trim a phrase each time a particular punctuation mark is met. The phrase is stored in the database with local drive.

It is important to remember that we use Standard Analyzer which performs stemming and stop word removal. The logic behind this has been to consider each phrase generated as a single document, so for each generated phrase we created a term frequency vector, like the code below:

```
if (tdocFreq.containsKey(term))
tdocFreq.put(term, tdocFreq.get(term).intValue() + 1);
else
tdocFreq.put(term, 1);
```

This data structure is a simple term frequency vector associating each term to be number of its occurrences in text. For example if we had:" Every man dies, not every man really lives.", the derived structure will be as shown in Table 1. It is related to the phrase and to keep track of the relation we associate each phrase through its own id and term frequency vector. This method associates the terms with their owns synonyms creating and filling another data structure, a JavaHashmap.

Table1: Term Frequency: Registered Document

Term	Synonyms
Every	2
Man	2
Die	1
Really	1
Lives	1

C. Searching on the index

In this step which is performed by the class SearchDocs, the index is questioned to retrieve documents containing terms also in the registered document. Plagiarism class permitting to query an index is IndexSearcher:

IndexSearcher is = new IndexSearcher(directory);

where directory is an object of class Directory containing the path of the index previously written.

Within SearchDocs two methods having the same name are declared: queryDocs; the first one takes only the structure with query pharse id associated to its term frequency vector (tdocFreq, Table1); the second method takes also the WordCount structure (Table2). In both cases we scanned this structure and for each query phrase it is operated in this way:

```
for each term
create a query
question the index
for each result obtained (for each document)
get the term frequency vector
fill two data structures:
- docVectors: document - vector
- docSumOfFreq: document - sum of the term frequencies
end for each
end for each
```

for each Document we took the respective term frequency vector, as [8]we specified to Index writer. We did this using methods getIndexReader and getTermFreqVector from class IndexSearcher, specifying id and field to retrieve.

TermFreqVector tfVector = is. Get Index Reader (). Get Term Freq Vector (Integer. parseInt (id),"contents");

We associated each retrieved document with respective term frequency vector in a data structure called docVectors.

docVectors.put(filename,tfVector);

Table2: Term Synonyms WordCount

Terms	Occurrences
Man	Adult male, homo, human being, human……
Die	Decease, perish, pass away, expire…….
Really	Truly, actually, in truth…
Lives	Survive, endure, exist, alive…

D. Evaluating Similarity with SCAM

Detecting plagiarism is not a simple string match; it should give a positive result for example by indicating either the registered document is a superset or a subset of the query document.Simple cosine similarity measure typically used is not enough to recognize overlap between documents; SCAM formula is a relative measure to detect overlap, irrespective of the differences in document sizes[2]. We have implemented the [18]SCAM formula to detect similarity among documents. This similarity formula re-turns a high value when the content of query document is either a subset or a superset of the registered document.

It is an alternative to the cosine similarity formula and it works on a set of words that are in common between test and registered document, a word w_i is included in the set C,

if the following condition is true:

$$\varepsilon - \left(\frac{fi(R)}{fi(T)} + \frac{fi(T)}{fi(R)} \right) > 0$$

€ is a constant value greater than 2; a large value of € expands the above set including words sometimes not relevant, a lower value of € reduces the ability to detect minor overlap, since some words can be excluded. fi(R) and fi(T) are the number of times wi occurs in registered documents (R) and query document (T)[5] . the score for the measures is given by:

$$S(T,R) = \frac{\sum_{wi \varepsilon C} fi(R) * fi(T)}{\sum_{i=0}^{N} fi(T) * fi(T)}$$

It returns the degree to which R overlaps T, normalized with the document T alone. The numerator works, as we said, on the frequencies of words in the pair of document from the set C. the relative similarity which is limited to the range 0 to 1, is given by:

similarity(T,R)= max[S(T,R),S(R,T)]

Where S(R, T) is the same formula with reserved operands as;

$$S(R,T) = \frac{\sum_{wi \varepsilon C} fi(R) * fi(T)}{\sum_{i=0}^{N} fi(R) * fi(R)}$$

Now we have all the information and data structures to be used in the implementation of SCAM formula. The code is below:

for each retrieved Document
get the terms
for each term
if the term appear in the test Document
get the test term frequency
get the term frequency
if condition EPSILON
calculate S(T, R)
calculate S(R, T)
end if
end if
end for
end for

E. Graphical User Interface

We developed a simple GUI allowing the user to accomplish two main tasks as indexing a dataset and checking for plagiarism. It has been chosen a windows interface, common and easy to understand; the screen shot in Figure 3 represents the main window. The index window includes

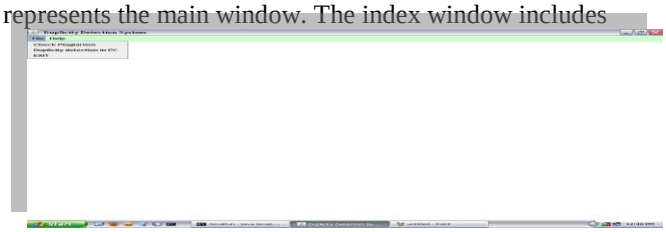

Figure 3: Scren Shot Main Window

the path of the directory containing the documents to line and sentence checking (figure 4). The list of the indexed documents and shown to the user. Figure 6 shows duplicity document in each local drive.

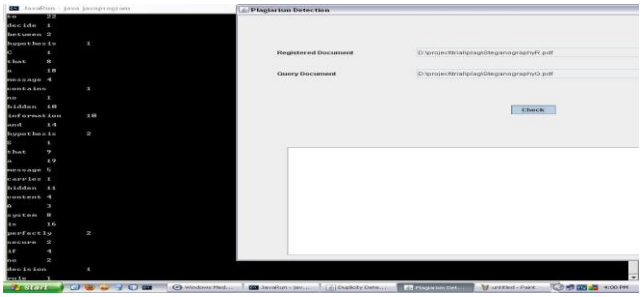

Figure 4: Comparison b/w registered and query document

We have calculated various similarity percentages on the basis of word occurrence frequencies, and line match frequencies Figure 5.

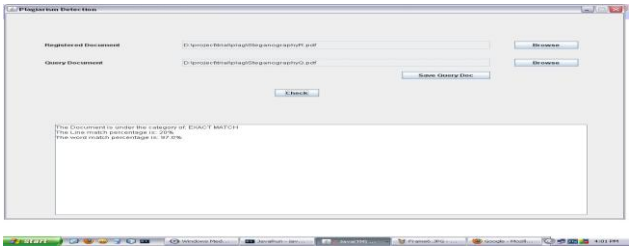

Figure 5: Result of plagiarism detection

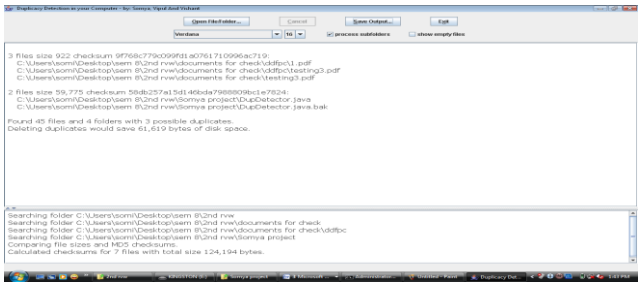

Figure 6: Duplication Detection in PC

IV. RESULTS

Here are some results calculated by considering a Query Document against the Registered Document.

We have calculated various similarity percentages on the basis of word occurrence frequencies, and line match frequencies. On the basis of results obtained the query document was categorized as either EXACT, PARTIAL, TRIVIAL, or NO Match with the Registered Document, and then further based on it, we can decide whether to add Query Document into the database or not figure 7.

Some results are as under:

Query Document: SteganographyQ.pdf
Registered Document: SteganographyR.pdf

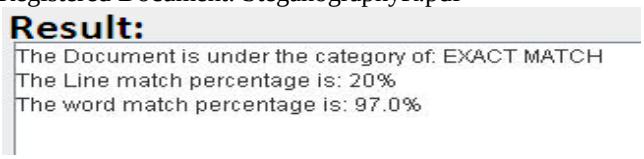

Figure 7: Exact Match

Query Document: SteganographyR.pdf
Registered Document: pdfFile.pdf

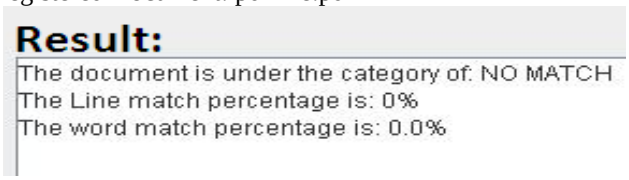

Figure 8: No match

Query Document: plag2.pdf
Registered Document: Plagiarism.pdf

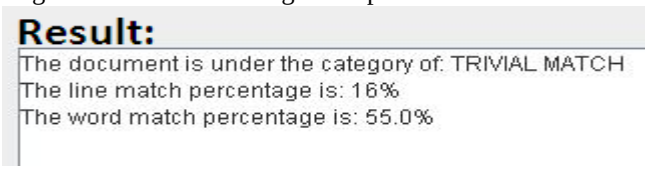

Figure 9: Trivial Match

V. ADVANTAGES

- The copy detection system provides a mechanism for displaying the location of overlap that exists between two documents. It shows the location of matching sentences for each pair of sentences in a comparison. Furthermore, the granularity of this overlap distribution can be adjusted to reveal more or less detail as desired.
- People who hire writers want to obtain unique and novel piece of writings for their magazines and newspapers. Such software helps the companies in publishing original articles in order to avoid law suits and other related problems.

VI. CONCLUSIONS

It shows (No, trivial, Partial, Exact) in our table3, we may be biasing our conclusions. For example, perhaps a threshold of 35% or 40% would give more desirable results, while in our table we have limited our threshold values to four. In spite of this, we only show four ranges because: (1) Showing more ranges would make it harder to visualize the results and (2) we believe (after analyzing the raw data) that these four ranges are adequate to roughly distinguish the various ca

Table 3: Difference for various matches

Classification Type	Word range%	Line range%
Exact Match	>=80	>=80
Partial Match	>=80	>=40
Trivial match	>=50	>=0.0
No Match	<50	<=0.0

REFERENCES

1. C. Justicia de la Torre, Maria J. Martn-Bautista, Daniel Sanchez, and Mara Amparo Vila Miranda.Text mining: intermediate forms on knowledge rep-resentation.
2. Manu Konchady. Building Search Applications:Lucene, Lingpipe, and Gate. Mustru Publishing, Oakton, Virginia, 2008.
3. Higher Education Academy ICS (Information and Computer Sciences) University of Ulster. Plagiarism prevention and detection. (n.d.). Retrieved March 17, 2010, from http: //www.ics.heacademy.ac.uk/resources/assessment/plagiarism/detectplagiarism.html.
4. Eduard Montseny and Pilar Sobrevilla, editors. Pro-ceedings of the Joint 4th Conference of the European Society for Fuzzy Logic and Technology and the 11th Rencontres Francophones sur la Logique Flouettses Applications, Barcelona, Spain, September 7-9, 2005. Universidad Polytecnica de Catalunya, 2005.
5. M. Bilenko, R.J. Mooney, W.W. Cohen, P. Ravikumar, and S.E.Fienberg, "Adaptive Name Matching in Information Integration,"IEEE Intelligent Systems, vol. 18, no. 5, pp. 16-23, Sept./Oct. 2003.
6. C. Sutton, K. Rohanimanesh, and A. McCallum, "Dynamic Conditional Random Fields: Factorized Probabilistic Models for Labeling and Segmenting Sequence Data," Proc. 21st Int'l Conf. Machine Learning (ICML '04), 2004.
7. V.S. Verykios, G.V. Moustakides, and M.G. Elfeky, "A Bayesian Decision Model for Cost Optimal Record Matching," VLDB J., vol. 12, no. 1, pp. 28-40, May 2003.
8. V.S. Verykios and G.V. Moustakides, "A Generalized Cost Optimal Decision Model for Record Matching,"Proc. 2004 Int'l Workshop Information Quality in Information Systems,pp. 20-26,2004.
9. N. Koudas, A. Marathe, and D. Srivastava, "Flexible String Matching against Large Databases in Practice,"Proc. 30th Int'l Conf. Very Large Databases (VLDB '04),pp. 1078-1086, 2004.
10. R. Agrawal and R. Srikant, "Searching with Numbers," Proc. 11th Int'l World Wide Web Conf. (WWW11), pp. 420-431, 2002.
11. W.E. Yancey, "Evaluating String Comparator Performance for Record Linkage," Technical Report Statistical Research Report Series RRS2005/05, US Bureau of the Census, Washington, D.C., June 2005.
12. W.W. Cohen and J. Richman, "Learning to Match and Cluster Large High-Dimensional Data Sets for Data Integration," Proc. Eighth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD '02),2002.
13. A. McCallum and B. Wellner, "Conditional Models of Identity Uncertainty with Application to Noun Coreference," Advances in Neural Information Processing Systems (NIPS '04),2004.
14. P. Singla and P. Domingos, "Multi-Relational Record Linkage," Proc. KDD-2004 Workshop Multi-Relational Data Mining, pp. 31-48, 2004.
15. H. Pasula, B. Marthi, B. Milch, S.J. Russell, and I. Shpitser,"Identity Uncertainty and Citation Matching," Advances in Neural Information Processing Systems (NIPS '02), pp. 1401-1408, 2002
16. S. Chaudhuri, V. Ganti, and R. Motwani, "Robust Identification of Fuzzy Duplicates," Proc. 21st IEEE Int'l Conf. Data Eng. (ICDE '05), pp. 865-876, 2005.
17. Project Gutenberg home page: http://www.promo.net/pg/.
18. Glatt Plagiarism Services home page: http://www.plagiarism.com/.
19. J. Brassil, S. Low, N. Maxemchuk, and L.O'Gorman. Document marking and identification using both line and word shifting. Technical report, AT&T Bell Labratories,2004. May be obtained from ftp://ftp.research.att.com/dist/brassil/docmark 2.ps.

AUTHORS PROFILE

Mr. Ranjeet Singh, Assistant Professor of SRM University, NCR Campus in IT Department. B.Tech in IT From UPTU in 2007 M.Tech in CSE From SRM University, Chennai in 2010 Currently doing Research in Database Security & Cloud Computing.

Mr. Chiranjit Dutta, Assistant Professor of SRM University, NCR Campus in IT Department. B.Tech in CSE From WBUT in 2008 M.Tech in CSE From MGR University, Chennai in 2010 Currently doing Research in Wireless Communication & Cloud Computing.