

# Study & Evaluation of Document Comparing Mechanisms

Anagha Gupte                      Rahul Lakhotia  
Department of Computer Science  
Duke University, Durham, NC  
{anagha, rahul}@cs.duke.edu

## ABSTRACT

Digital libraries have made access to documents very easy but this also makes documents vulnerable to being copied. The illegal distribution of documents discourages authors/news feed services to share their information. Hence it is extremely essential to protect intellectual property. In this paper we have looked at the earliest comparison methods, namely string comparison algorithms that compare text strings. We have then looked at different methods used to tackle the problems of plagiarism, like copy prevention and copy detection. Our search has led to the discovery of several good document comparison techniques and we've evaluated three of these that we found to be the best. They are COPS, SCAM and CHECK. We've proposed enhancements to these and hope to implement these enhancements.

## 1. INTRODUCTION

Today's digital age has allowed papers on various different topics to be posted into digital libraries and these are accessible through the Internet. They have been freely accessible for all users. Availability and distribution of this kind of information is extremely useful as it gives quick and reliable access to material on several topics. But it also comes with certain disadvantages. It can lead to illegal distribution of information or plagiarism.

People have tried to overcome these difficulties by Copy Prevention and Copy Detection mechanisms.

Copy prevention involves methods like storing intellectual property on stand-alone servers, encrypting information; creating secure computer networks etc. this approach actually places physical restrictions on the distribution of documents. It is a very pessimistic approach and it proves a hindrance to honest users.

Copy Detection involves physically detecting copies of documents, that is, comparing documents using different techniques to detect the similarity between them. It is a more optimistic approach than copy prevention. This is the focus of our research.

Our attempt is to research techniques that, if given two documents, detect the similarity between them. Two documents can be similar in more than one ways. They could be exact copies of each other or almost similar. One document can be an older or a newer version of the other or it could be that one paper can be a journal/conference version of the other.

After a research of several document comparison methodologies, we have understood the underlying mechanisms of each one of them and essentially how they function. This further led to an exhaustive study of some of the best methods and comparisons between them. We have tried to enhance some of these methods and we plan to employ our enhancement ideas into future work along this area.

The following section describes some related work in the area. Sections 3 and 4 describe copy detection mechanisms of COPS and SCAM, while the following section compares the two. Section 6 describes the fastest mechanism we

have found: CHECK. Section 7 gives the enhancements we propose to do and we conclude in Section 8.

## 2. RELATED WORK

We have surveyed existing approaches of comparing two documents.

Matching textual information begins at the lowest level with simple string matching. If two strings can be compared and found same, we can see where there is a copy of the text. In the past, several algorithms have been proposed for determining exact string matches. We have surveyed these and analyzed their running times just as a basis for our later survey. We did this to get an idea of how text-comparing methodologies first developed and progressed.

Below we have discussed some string matching algorithms. The later ones are an enhancement of the earlier ones, but however much these are improved upon, their scope is limited to matching exact strings, character by character. There is no notion of matching similar information or deriving a meaningful relationship between the texts in two documents. Here are some of the important string matching algorithms:

### BRUTE FORCE ALGORITHM

This approach is simply matching  $x$  (our given string) with substrings from  $y$  at successive positions in the text string.  $x$  is compared with substrings  $y(l, l+m-1)$ , until a match is found or the end of the text is reached. It requires the input text string to be buffered since backtracking in the text is required in the event of an unsuccessful match. In the worst case, this method runs in  $O(mn)$  time, but for practical applications like searching English text, the expected performance is  $O(m+n)$ .

### KNUTH- MORRIS- PRATT ALGORITHM

This algorithm is an improvement on the Brute Force algorithm in that it avoids backtracking, which was essential in Brute Force. In the event of a mismatch, this algorithm doesn't employ backtracking, but instead it takes advantage of

known information. The text string is processed sequentially from left to right. When a substring match attempt fails at symbol  $x_j$ , say, the previous  $j-1$  text symbols will be known since they will be given by  $x(1, j-1)$ . This fact can be exploited to determine how far to shift the pattern to the right for the next match attempt. An auxiliary next table containing this shift information is computed in  $O(m)$  time from the pattern before searching the string. Though this method performs well for highly self-repetitive patterns and text, this situation occurs only rarely in actual applications. So, in practice, the Knuth- Morris- Pratt algorithm is not likely to be significantly faster than the Brute- Force approach. It has the only advantage of not having to backtrack in the input text, obviating the need for involved buffering- schemes when reading from sequential text streams, for example. It runs in worst case time  $O(m+n)$

### BOYER MOORE ALGORITHM

In the Boyer Moore search algorithm, the pattern is scanned across the text string from left to right but the actual symbol comparisons between the pattern and the texts are performed from right to left. The first comparison is therefore between  $x_m$  and  $y_m$ . In the event of a mismatch, if  $y_m$  does not occur at all in  $x$ , then the pattern may be shifted safely ' $m$ ' places to the right. This is due to the fact that the possibility of  $x$  occurring in  $y$  starting at any of the first  $m$  positions has been ruled out. The next comparison would then be between  $x_m$  and  $y_{2m}$ .

The speed of the Boyer Moore algorithm is achieved by eliminating portions of the text string that cannot possibly result in a successful match. For a large alphabet and small pattern, the expected number of symbol comparisons is about  $n/m$ , and is  $O(m+n)$  in the worst case. This approach is much faster than the Knuth- Morris- Pratt approach.

These approaches for string searches can have applications like keyword searching for text editors and word processors, but their scope may be limited to just that. They are not scalable to large documents. More intricate techniques are used for comparing whole documents, which

involve a lot more than simply matching pieces of strings with each other. There is a matching of semantic information involved and that requires the development of intricate methodologies.

## **COPY PREVENTION**

In the past, attempts to completely eradicate copying of documents have been made. There is an approach called 'secure printer' where the document provider first ensures that the user is authorized to access a document. Another approach is 'active document', where a user receives a document generation program instead of the actual document. Also a confirmation is desired from an information provider whenever the program is being executed. This ensures access of information to only authorized users.

As we can see, these approaches are very pessimistic in nature. They seem to be very time consuming and are designed with a pre-conceived notion that most users are dishonest.

## **COPY DETECTION**

Better than these are the Copy detection mechanisms. These mechanisms provide a slightly more optimistic approach to detecting similarities between documents. Users are openly allowed to access documents. However, illegal usage of the intellectual property is detectable by two kinds of methods- the Signature and the Non signature approach.

In the Signature approach, a 'signature' or unique identifying key is added to each document, which is used to trace the origins of a document. This approach too has certain drawbacks. Since the signature provides as the only identifying key on a document, it is the only thing used while checking for document similarity, this means that if the signature is removed, the document becomes no more traceable. Removing the signature is easy and can be done automatically to prevent from copy detection. Also, this signature comparison method will only detect complete overlaps of documents. Hence it is useful only when two documents are exactly alike.

All the above reasons and the disadvantages of the above methods prove that copy detection techniques generated to compare two documents are the most useful. They compare the semantics of documents rather than merely their strings. They can be scalable to larger documents. We evaluated a number of these and have decided to focus on COPS (Copy Protection System), SCAM/ dSCAM (Stanford Copy Analysis Mechanism/ distributed Stanford Copy Method) and CHECK. We have evaluated and compared these.

## **3. COPS**

COPS is a copy Protection System that can be employed for detecting similar documents. The underlying idea of this system is that documents are registered in a database. New documents that have to be compared with existing ones are either compared against the existing database or first registered and then compared.

COPS uses the idea of an OOT (Ordinary Operational Test). We will first give an overview of OOTs and then talk about the implementation of COPS.

### **3.1 ORDINARY OPERATIONAL TEST (OOT)**

A document can be divided into well defined parts, consistent with the underlying structure such as sections, paragraphs, sentences, words or characters. Each of these divisions is known as a unit type and a particular instance of the unit type is called a 'unit'. OOTs use the term 'chunk'. Chunk is a sequence of consecutive units in a document of a given unit type. Chunks can be of different sizes, they can overlap and need not cover the document completely. For example, in a unit ABCDEFG, chunks can be A, B, C, D, E, F, G or AB, BC, CD, DE, EF, FG. The method of selecting chunks is called a 'chunking strategy'. OOTs use hashing to detect matching chunks. It uses the following set of procedures in general:

```
PREPROCESS(R, H)
  CREATETABLE (H)
  For each r in R INSERT(r, H)
```

```

INSERT(r, H)
  C= INS-CHUNKS(r) /*OOT dependent
    for each <t,l> in C
      h= HASH(t)
/*assume size of reg. Doc. May be obtained
from id*/
  INSERTCHUNK(<h,r,l>,H)
/* implementation unspecified*/

```

```

DELETE(r, H)
  C= INS-CHUNKS(r)
  for each <t,l> in C
    h= HASH(c)
    DELETECHUNK(<h,r,l>, H)
/* implementation unspecified*/

```

```

EVALUATE (d, H)
  C= EVAL-CHUNKS(d)
  SIZE = |C|
  MATCHES = {} /* empty set*/
  for each <t,l> in C
    h= HASH (t)
    SS= LOOKUP(h,H)
/*returns all <r,l> with matching h */
  for each <r, lr> in SS
    MATCHES += <|t|, ld, r, lr>
  return DECIDE(MATCHES, SIZE)
/*OOT dependent*/

```

The PREPROCESS(R,H) operation takes R registered documents and creates a hash table H. The INSERT procedure inserts documents in a hash table. It uses a function INS-CHUNKS(r) to break up a document into chunks. It returns a set of tuples <t, l> where each tuple corresponds to one chunk of the document r. t is the text of the chunk and l is the chunk location. Every tuple has an entry in the hash table.

The procedure DELETE is similar to insert. It destroys a chunk, that is, a tuple <t, l> and removes its entry from the hash table.

The procedure EVALUATE tests a document for violations. It uses a chunking function to break up a document. Once chunking is complete, the function searches for chunks in the hash table and produces a set of tuples called MATCH. These tuples represent matches found in the hash table. This set MATCH is then

passed into a function called DECIDE(MATCH, SIZE) to return the set of matching documents. If this set is non- empty, then it shows that similarity has been found in two or more documents.

In the implementation of COPS, registered documents are stored separately. So, similar documents are returned to the user with the matching chunks highlighted.

COPS has the following modules:

- TEX- ASCII converter
- DVI- ASCII converter
- Troff- ASCII converter
- Sentence identification and hashing
- Document registration
- Query processing
- The database



Modules in COPS

When a new document arrives, it is either entered into the database or tested against existing documents. COPS takes documents of different formats like TEX, DVI and troff and converts them into ASCII format, before they are broken up into sentences and hashed in a table. Hashing essentially means hashing the document's individual sentences. Every document registered into the database is assigned a unique ID. So, the ID is stored against each sentence belonging to that particular document.

COPS implements several types of chunking strategies like INS-CHUNKS and EVAL-CHUNKS and also different decide functions like the match-ratio function or ordered matches. There can be a mix and match of different chunking and decision functions and they would produce different results. Hence, often, an educated guess has to be made about the documents before choosing the functions to be used.

COPS, as a first approximation identifies sentences by taking words up to a period or a question mark. This will often create one word or one letter sentences due to abbreviations like 'i.e.' or U.S.A. and it will flag incorrect matches. These trivial problems can be overcome by disregarding single word sentences.

A general overview of the chunking strategy of COPS is as follows:

#### COMBINE (N- UNITS, STEP, UNIT- TYPE)

This procedure does the chunking of a sequence, where N- UNITS is the number of units to be combined into the next chunk, STEP is the number of units to advance for the next chunk and UNIT- TYPE indicates what should be considered as a unit, that is whether one unit should be a word or a sentence.

From the above, we can see that COPS is using a sentence-based comparison in order to detect similarity between documents.

### 3.2 DRAWBACKS OF COPS

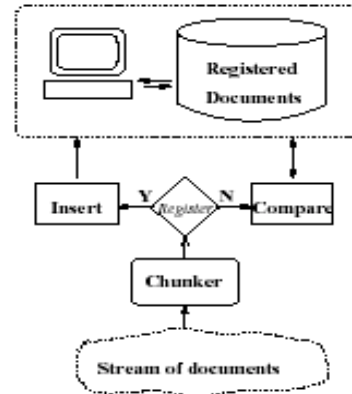
The first basic drawback of this is that it has difficulty in detecting sentences. Since it breaks up sentences based on locations of periods or question marks, it often makes mistakes. For example, the sentence, "We live in the U.S." may be wrongly broken up before the S. though COPS may eliminate standard abbreviations or one-word sentences, yet it is not a foolproof way of detecting sentences.

Another important drawback of COPS is that while it checks for sentence overlaps, it involves random probes into the database and this may be expensive. COPS wont detect partial overlaps and hence the word-based schemes are better than it.

COPS also lacks the ability to take in postscript files and detect paragraphs, graphs and equations. These are simply eliminated during the ASCII conversion step, and this might result in loss of vital information.

## 4. SCAM

This copy detection mechanism works on the comparison of the relative frequency of the words presenting a document. The basic architecture of a registration server with a database as a repository is given in the figure below.

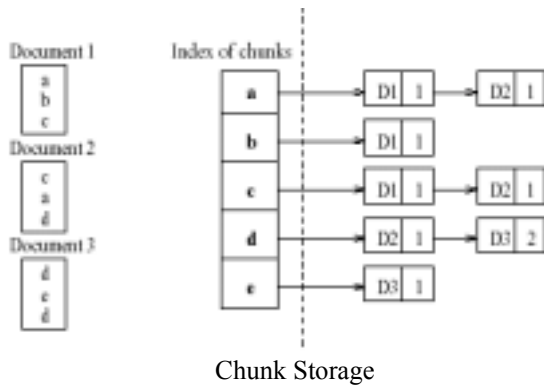


Architecture of SCAM

The chunker takes in documents and breaks them into "chunks" of sentences or words. The comparison process consists of comparing these chunks. Once the document is found to be original, they are then stored in the database for future reference.

The storage of these chunks is very important as it has an influence on the time taken for comparison purposes. SCAM uses an Inverted Index structure, as used in traditional IR systems. A chunk in SCAM is a word. An index of all the chunks (words) present in the vocabulary of the documents is maintained. Each chunk has a list of *postings* associated with it. Each posting consists of the two values. The document ID that the chunk appears in and the frequency with which it occurs in that particular document.

During the comparison process only those chunks that are present in the document to be compared are looked up.



A word about the size of chunks: The bigger the size of the chunks the less the probability of matching unrelated documents. This is so because two documents having same sentences (albeit very general statements) would be deemed similar when chunks are sentences. If chunks are paragraphs, they would not show as same due to the other mismatches in the paragraph. However, a large sized chunk also means it is more difficult to match related (but not exactly similar) documents. This can be seen in the case where two paragraphs have 6 sentences each with 5 same ones. Here if the chunk is a paragraph, then matching would not occur while if a chunk is a sentence, 5 out of 6 would match giving a high percentage of similarity.

Storage space required for the chunks also increases as size of the chunks increases. This is so because the number of words does not increase after a certain point. However the number of ways these words can be arranged keeps increasing as the size of the chunks increasing. Thus having a larger chunk would require more memory and eventually more disk accesses.

Although using words as chunks is beneficial we cannot use the same comparison process (vector space model) as used for sentences. This is because a high similarity of the words used in two documents does not necessarily mean they are similar.

SCAM employs the relative Frequency model for comparison purposes. In this it first computes the *closeness* set of the two documents. This consists of the list of all the words that occur the same number of times in

the two documents. This is given by the following formulae:

$$\epsilon - \left( \frac{F_i(R)}{F_i(S)} + \frac{F_i(S)}{F_i(R)} \right) > 0$$

The  $\epsilon$  is added to give tolerance level so that the user can have a parameter to adjust the level of similarity of the closeness set. A high value of  $E$  increases the chances of false positives but takes into account more words for comparisons. A low value of the  $E$  lowers the chances of false positives but also loses the ability to detect minor overlaps.

From this closeness set we compute the subset of the two documents based on the following formulae:

$$subset(D_1, D_2) = \frac{\sum_{w_i \in C(D_1, D_2)} \alpha_i^{F_i(D_1)} F_i(D_2)}{\sum_{i=1}^N \alpha_i^{F_i(D_1)}}$$

Intuitively, this measure is computing the asymmetric subset measure, with only the close words. The similarity between two documents is then defined as in the following way:

$$sim(R, S) = \max\{subset(R, S), subset(S, R)\}$$

Thus we have a high measure for those documents that are either a subset or a superset of the other document. SCAM uses this to determine if documents are copies of one another or not.

#### 4.1 DRAWBACKS OF SCAM

It is a word-based scheme, so it is better than a sentence based scheme, since it can detect partial overlap in sentences. But the method has a very high number of false positives. This means that a number of results that are not actually similar are reported as similar.

It is required to choose a value of  $E$  while detecting an overlap. According to currently implemented methods, this is done empirically, and seems to be giving fair results. However, it has to be able to work for a variety of documents and hence it is important to find a good value of  $E$ . A scheme needs to be devised to determine this based on heuristics.

Lastly, most documents are scattered around the world in different databases. Having them all in

one place is not possible. However, a more distributed approach can be implemented in which the servers are dynamically queried during comparison.

## 5. COMPARISONS

A comparison of SCAM and COPS shows that the SCAM approach is an improvement over the latter. The comparison has been done in two ways. First we see how the two ways differ when reporting on similarity. Table 1 gives this result.

Here we see that both mechanisms report a similar percentage for a large number of documents (approx 99%). However, there are significant differences in the documents they report this overlap in. As shown, Scam shows 2 documents to have only 21-30 % overlap while COPS shows them to have 81-90 %.

The second comparison sees which of the mechanisms is correct for these cases. Table 2 and Table 3 give these results. The table is simplified a bit so that comparisons that yielded less than 33% similarity are stored as NONE. Those between 33% and 67% are stored as SOME. Those between 67% and 90% are stored as LOTS while those above 90% as FULL.

We see the better performance of SCAM. If we have a threshold value of 33% to determine whether a document is plagiarized or not we see that of the 249 plagiarized documents, COPS reports 43 % to be not plagiarized while SCAM reports only 2% of them in a similar manner. Also for detecting exact copies with a threshold of 90% COPS would not be able to detect any of the 38 documents as copies, while SCAM would have a 100% rate.

This better performance however comes at a slight cost. SCAM also reports a larger number of false positives. As seen in table three, for a threshold level of 33% COPS reports that none of the 295 documents were plagiarized. This is as it should be. However SCAM reports 100% of them to be plagiarized. This number is very high as we are only considering the document pairs that we know to have errors. However, if

the entire picture is taken in to account then the number is much lower due to the large total number of documents (i.e. 1,520,289 documents).

In summary, one can say that COPS misses a number of cases of overlap. SCAM is better at detection but has a number of false positives. Depending upon whether we want a more automated process or a more accurate process we can choose between COPS and SCAM. With SCAM a human would have to parse through the results to remove the false positives.

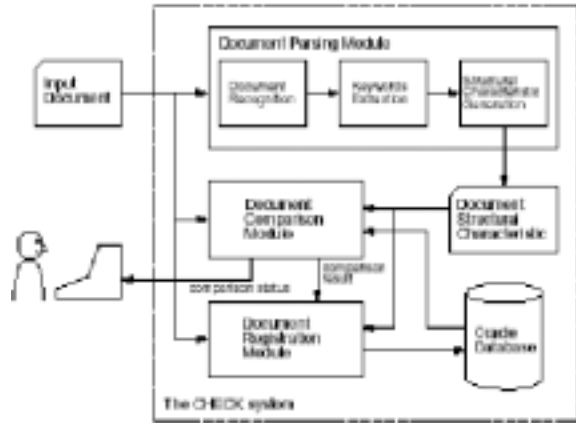
In the table 1 we saw that a very large number of the documents compared had 0% similarity. The third method CHECK takes advantage of this very feature and tries to eliminate comparisons between such documents at the start.

## 6. CHECK

This technique tries to improve over the other two by eliminating extra unnecessary work. The basic notion is that documents that are of different subject matters do not need to be compared. Copying between such documents would be very rare and if present would not contribute to the matter being discussed in the document. As mentioned, this is a “registration-based” mechanism.

### 6.1 ARCHITECTURE:

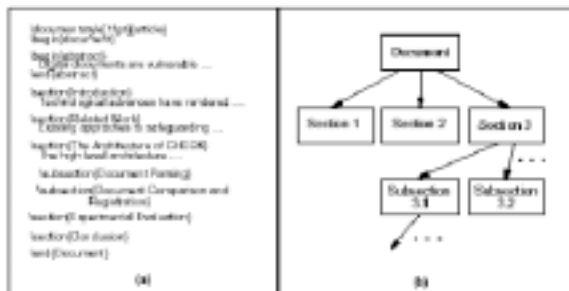
The technique basically consists of three modules: document parsing, document comparison and document registration whose interrelations are shown in the figure. The document comparison compares two documents for similarities, while the DP converts a document into its Structural Characteristic and extracts its key words. The Structural Characteristics’ of various documents are compared in pairs, in the document comparison module, and based on a minimum threshold level of similarity documents are designated as copies or originals. Lastly, the document registration module takes trusted original documents and registers them in a database for later use. We now describe each of the modules.



Architecture of CHECK

## 6.2 DOCUMENT PARSING:

This module has three functions. The document is converted into its ASCII format. From this a tree like structure called the document tree is built. The nodes of this tree are the various sections, subsections etc. This thus gives the layout of the document, depicting how sections are arranged and which subsections are parts of which sections. Please see the attached figure. This division is done till the granularity of a paragraph per node. The present module is able to do this only for LATEX documents, but since this is the common format in which papers are written, we feel this is sufficient. Similar techniques can easily be developed for documents/papers in other formats. During this pass over the documents, the input words of the document are converted to their canonical form by an intelligent lexicon.



Document Tree Structure

The second function of this module is Keyword extraction. All the words in the document are divided into two types. Open class and closed class. Open class words are descriptive words

that have meaning, while closed class words are the frequent ones used for sentence construction and usually are not meaning bearing. From the words that are classified as open class, we take the ones occurring within one standard deviation of the mean occurrence of all the open class words. These words are the keywords. One can also employ heuristics and give more importance to words that are **bold faced** or in *italics*.

The final function of this module is the generation of the Structural Characteristic. The Structural Characteristic is a combination of the document tree and the keywords. For each node in the tree, we find the set of keywords that best describe the node and assign these with higher weights. This thus describes the document in multiple levels with the necessary importance given to the appropriate words in each level/node. For example in the Root node the weight of each keyword will be given by:

The nodes representing the sections, subsections etc have only those keywords that are present in those sections subsections etc. The weight of the keywords are given by:

This weight age system gives due importance to the key words based on the sections they are present in.

## 6.3 DOCUMENT COMPARISON

This module takes the Structural Characteristics of the two documents in question and starts comparing them at the root. Since the information of each document is now in similar format (Structural Characteristic) it is easier to compare the documents. The nodes at each level are compared. If they appear similar past a certain threshold then their children are further investigated. On the other hand, if the words do not match, then no further comparisons are needed. If the investigation reaches the last node (representing a paragraph), then a final sentence by sentence comparison can be done to determine exact copies.

The scoring mechanism used to compare the documents is twofold. We first *normalize* the

vectors used to represent the particular nodes' keywords and weights. This is done as different documents can have a different number of keywords and comparing such vectors is difficult. Normalization is achieved by generating a vector with the union of the two vectors that need to be compared. This union vector with appropriate weights given to the elements is used as the normalized vectors. The weight of a keyword in the normalized vector of a node is zero if the keyword is not present in the original vector of the node. Otherwise, it is a value between zero and one based upon the weight age of the keyword in the union vector in the context of concerned document.

Once we have the normalized vectors, the similarity between them can be easily computed by taking the *dot product* of the two vectors.

This dot product gives the cosine of the angle between the two vectors. As the angle decreases, the cosine values increase, and thus more is the similarity. If this value crosses a threshold value we say the vectors are similar and carry on to investigate the nodes lower down the tree in a similar fashion. This threshold level can be manipulated for the degree of similarity sought.

The comparison process at a particular level involves comparing all nodes with each other so that documents where the sections have simply been rearranged are also detected. This process saves a lot of time by eliminating a lot of the unproductive work.

#### **6.4 DOCUMENT REGISTRATION**

This module simply takes the documents given by the user and stores them in the database as originals, to be used as reference documents. It should be noted that this authority should only be given to the administrators so as not to compromise the authenticity of documents. When a particular document has passed through the verification (comparison) process and is found to be an original the document registration module stores it in the database for future reference too.

#### **6.5 DRAWBACKS OF CHECK**

We have seen that all of the above document comparing methods involve registering of the documents into a database. If a new document comes in, it may not be registered, but it is still compared against the set of registered documents. This means that without registering, no comparisons can be done. This is definitely a drawback since it involves convincing authors of all documents to register their respective documents. This might involve taking into account a lot of time and space, which might not be feasible. All this has to be checked before making these techniques global. Also each has its own drawbacks as we may have realized in the earlier sections.

CHECK seems to be an improvement on the earlier two methods, but it has its own drawbacks. CHECK works right now for only LATEX documents. It is true that almost all papers today are written in LATEX. But you never know when the formatting of a document maybe different. So we think that this technique must in some way be extended to other formats too.

When a new document arrives, that is not previously found in the db, it is compared with existing ones, and if no match is found, then it is saved and registered as a new document. Now supposing this document was actually copied from some other original document, we would not be able to detect this. Moreover, since this copied document is registered in the database, if we later come across the original document, it will be declared as a copy.

The comparisons done in the lower levels where there are a number of nodes at each level can become very high. The comparison process thus becomes very long and complex.

#### **7. ENHANCEMENTS**

Based on the above study, we feel that CHECK is the best method. Our enhancements thus deal with improving this method to further solve the problems we have mentioned earlier.

All of the methods above do not consider graphs and figures. This is mainly because it is very difficult to co-relate them. Graphs for different things can be similar and thus finding out that they are same does not mean anything. However, with the use of the CHECK method, we can now store the graphs and figures in their respective nodes. When these nodes are found to have similar information in the manner discussed above, we can then compare the graphs present in these nodes. Now since we know them to have similar information then the similarity in the figures and graphs will also add to our confidence of similarity.

Also a separate node with information about the document version and conference information can be stored. In cases where we want to find about version history and conference/journal papers then only these nodes need to be compared.

Lastly, CHECK only works for LATEX documents at present. A module for taking in other formats like DVI and troff need to be developed too.

## 8. CONCLUSION

We have seen in our paper that several different methods have been implemented for the comparison of digital documents. The main purpose of our comparison is to detect similarities within documents. This is helpful to see whether documents are exact matches of each other, or if plagiarism has occurred. This can further be extended to detecting older/ newer versions of documents or journal/ conference versions of documents and this is our main goal in the future. We have seen that the simplest comparison methods began with string matching algorithms, which simply compared a given pattern to strings in a document to see which ones matched. These had a limited scope and were not scalable.

Copy Protection System (COPS) is mechanism that was implemented in 1994 which is a more enhanced mechanism that compares documents based on sentence comparisons. We have seen the drawbacks of COPS. SCAM (1995) seems to

be a further enhancement of COPS since it does a word-based search as opposed to a sentence based search, so attempting to do away with the drawbacks of COPS. The CHECK method (1997) is a further improvement of SCAM. It eliminates comparisons of unnecessary documents, thus being a more efficient and effective method. We have seen the trend in the improvement of document comparison methodologies and we think it is safe to say that any kind of that we make should be done on CHECK, since CHECK itself is an improvement over the other two methods that we've talked about. So we should use CHECK as a basis in order to improve comparison methods. Our project comprised of researching and reviewing existing methods as a basis to do future work on document comparison. We hope to continue this project in the future by eliminating the drawbacks of existing methods and implementing our proposed enhancements.

## 9. REFERENCES

- [1] Stephen, Graham A. String Searching Algorithms
- [2] C.W. Johnson, B.D. McKay and V Sharma, Further methods for detecting plagiarism in student programs, Australian Computer Science Communications, 9 (1987)
- [3] plagiarism.org- this site is concerned with problems of plagiarism over the internet and has several useful links.
- [4] Krisztian Monostori, Arkady Zaslavsky, Mainz Schmidt Document Overlap Detection System for Distributed Digital libraries
- [5] Sergey Brin, James Davis, Hector Garcia-Molina Copy Detection Mechanisms for Digital Documents
- [6] Narayanan Shivakumar, Hector Garcia-Molina SCAM: A Copy Detection Mechanism for Digital Documents
- [7] Hector Garcia, Luis Gravano, Narayanan Shivakumar dSCAM: Finding Document Copies Across Multiple Databases
- [8] Antonio Si, Hong Va Leong Rynson W.H. Lau CHECK: A Document Plagiarism Detection System.

	COPS	0	1-10	11-20	21-30	31-40	41-50	51-60	61-70	71-80	81-90	91-	100
0	1,518,037	1,515,333	15	40	6	1518	618	297	121	45	25	2	17
1-10	68	0	3	2	0	19	9	18	4	10	1	2	0
11-20	48	0	0	0	12	2	3	8	6	1	2	2	12
21-30	57	0	0	0	1	20	9	4	2	7	3	5	6
31-40	38	0	0	1	9	1	3	2	6	4	4	3	5
41-50	32	0	0	0	4	3	5	1	3	2	6	2	6
51-60	33	0	0	0	3	4	6	3	2	2	2	1	10
61-70	15	0	0	0	4	0	1	1	0	2	1	0	6
71-80	9	0	0	0	1	1	0	0	0	0	6	0	1
81-90	<b>64<sup>a</sup></b>	0	0	0	2	2	5	1	2	1	3	<b>39<sup>b</sup></b>	<b>9<sup>c</sup></b>
91-99	29	0	1	0	0	0	1	1	1	0	0	10	15
100	1859	0	0	1	0	4	0	0	0	0	0	7	1847
	Total	1,515,333	19	44	42	1574	660	336	147	74	53	73	1934

Table 1: Comparison of COPS results with those of SCAM

Test	System	# Satisfy(Test)	None	Some	Lots	Full
PLAGIARIZED	COPS	<b>249<sup>c</sup></b>	<b>43.78<sup>d</sup>%</b>	26.91%	<b>26.91<sup>e</sup>%</b>	2.41%
	SCAM	249	2.01 <sup>g</sup> %	38.55%	26.91%	<b>32.53<sup>h</sup>%</b>
SUBSET	COPS	120	17.50%	35.83%	45.83%	0.83%
	SCAM	120	0.83%	2.50%	30.00%	66.67%
COPIES	COPS	38	5.26%	2.63%	92.11%	0.00%
	SCAM	38	0.00%	0.00%	0.00%	100.00%
RELATED	COPS	253	44.66%	26.48%	26.48%	2.37%
	SCAM	253	1.98%	39.92%	26.48%	31.62%

Table 2: Detection test of SCAM and COPS

Test	System	# -Satisfy(Test)	None	Some	Lots	Full
PLAGIARIZED	COPS	<b>295<sup>i</sup></b>	<b>100.00<sup>k</sup>%</b>	0.00%	0.00%	0.00%
	SCAM	295	0.00%	<b>87.12<sup>l</sup>%</b>	10.17%	2.71%
SUBSET	COPS	424	90.33%	5.66%	2.83%	1.18%
	SCAM	424	0.94%	82.55%	14.39%	2.12%
COPIES	COPS	506	79.45%	13.04%	6.32%	1.19%
	SCAM	506	0.99%	69.76%	19.17%	10.08%
RELATED	COPS	291	100.00%	0.00%	0.00%	0.00%
	SCAM	291	0.00%	86.60%	10.31%	3.09%

Table 3: False Positive Test Comparison of COPS and SCAM