In Proceedings of ACM Symposium for Applied Computing, pp. 70-77, Feb. 1997.

CHECK: A Document Plagiarism Detection System

Antonio Si Hong Va Leong Rynson W. H. Lau
Department of Computing
The Hong Kong Polytechnic University
Kowloon, Hong Kong
{csasi, cshleong, cswhlau}@comp.polyu.edu.hk

Keywords: Digital Libraries, Copy Detection, Document Plagiarism, Information Retrieval

ABSTRACT

Digital documents are vulnerable to being copied. Most existing copy detection prototypes employ an exhaustive sentence-based comparison method in comparing a potential plagiarized document against a repository of legal or original documents to identify plagiarism activities. This approach is not scalable due to the potentially large number of original documents and the large number of sentences in each document. Furthermore, the security level of existing mechanisms is quite weak; a plagiarized document could simply by-pass the detection mechanisms by performing a minor modification on each sentence. In this paper, we propose a copy detection mechanism that will eliminate unnecessary comparisons. This is based on the observation that comparisons between two documents addressing different subjects are not necessary. We describe the design and implementation of our experimental prototype called *CHECK*. The results of some exploratory experiments will be illustrated and the security level of our mechanism will be discussed.

1 INTRODUCTION

Technological advances have rendered digital libraries a concrete possibility. Fueled largely by the gaining popularity of World Wide Web servers and browsers such as Netscape, free resources and information are readily available on the Internet. However, the same enabling technologies also increase the threat of illegal copying and distribution [3]. Current technologies do not provide proper safeguard on intellectual properties. Consequently, information providers tend to provide valuable information through a closed system such as the current CD-ROM system provided by IEEE to distribute its publications.

With the new technology, it is important to allow users to access information freely over the information superhighway, while at the same time, restrict them from illegal copying and distribution of information. One approach to address this issue is to provide a copy detection service through which legal original documents could be registered, and copies can be detected [3]. Most existing copy detection mechanisms employ an exhaustive sentence-based comparison method in comparing a potential plagiarized document against all registered documents to identify plagiarism activities. This approach is not scalable due to the potentially large number of registered documents and the large number of sentences in each document. Furthermore, the security level of existing mechanisms is quite weak; a plagiarized document could simply by-pass the detection mechanism by performing a minor modification on each sentence.

We note that comparisons between documents addressing different subjects are usually not necessary in copy detection; this is because it will serve no purpose to copy materials from a document describing unrelated subjects. Yet, the majority of comparisons belong to this category. In this paper, we propose a copy detection mechanism that will eliminate most unnecessary comparisons. When a potential plagiarized document is compared against a registered document, we employ information retrieval techniques to pre-process the documents to determine the semantic meanings of the documents. If the documents are on different subjects, further comparisons between the documents could be eliminated. This rationale is recursively applied to individual organizational constructs of different granularities including sections, subsections, subsubsections, and paragraphs. In other words, we recursively apply information retrieval techniques to each individual organizational construct until we find two paragraphs which are highly related semantically. The paragraphs are then compared in detail, i.e., on a per-sentence basis, to determine if the paragraphs overlap in a substantive way. By studying the semantics of the documents in additional to their syntax, an extra level of security is provided because users could no longer by-pass the detection mechanism by simply performing a minor modification to each sentence, such as changing the tense from present to past, as the semantics of the documents would very likely remain unchanged.

The remainder of this paper is organized as follows. In Section 2, we survey existing work on safeguarding intellectual properties. The design and implementation of our experimental prototype called *CHECK* will be

described in Section 3. In Section 4, we discuss the results of some exploratory experiments and study the security level of our mechanism. Finally, we present a conclusion of the work described here and some future research directions in Section 5.

2 RELATED WORK

Existing approaches to safeguarding intellectual properties can be roughly categorized into copy prevention [6, 12] and copy detection mechanisms [2, 3]. Copy prevention mechanisms follow a pessimistic approach in which access to information is only restricted to a few authorized users; their aim is to restrict the distribution of documents as much as possible.

One approach to copy prevention is based on the notion of a *secure printer* [12]. When a user requests a document from an information provider, the provider first ensures that the user is authorized to access the document. The document is then, encrypted with a public key. The encrypted document is sent to the authorized user, who could then, print it by a dedicated printer, equipped with the corresponding private decryption key. Users with no access to the secure printer will not be able to print out a stolen encrypted document. The limitation of this approach, however, is that users cannot preview a document for suitability.

Another approach to copy prevention is based on the idea of *active document* [6]. Instead of sending the document to a user, the information provider sends a document generation program, which can generate the desired document. In addition, the program also requests confirmation from the information provider whenever it is being executed. This allows the information provider to ensure that users of the program are indeed authorized. A major drawback to this scheme is that one can simply run the program within an emulated environment [3].

Notice that the purpose of copy prevention is to ensure that the user receiving the document is authorized. Once a user obtains the legal copy of a document, the issue of re-distributing copies of the original document is not addressed. Furthermore, the issue of plagiarizing a portion of the original document could not be addressed by copy prevention mechanisms.

Unlike copy prevention mechanisms, copy detection mechanisms are optimistic. Users are assumed to be honest and are allowed to access documents under an open environment. However, detection techniques are employed to identify illegal copying and distribution activities.

One approach to copy detection is to incorporate a *watermark* into a document [2]. This watermark identifies the original user who requests the document. If the document is copied electronically, i.e., the document is possessed by a person other than the original user, a violation has occurred, and the watermark can witness the culprit. The major weakness of the watermark scheme is that the watermark itself can be destroyed by processing the document through a lossy compression operation like JPEG [8]. In addition, detection of partial copying is not possible. Watermarking is a kind of passive copy detection approach.

Early active copy detection systems mainly deal with plagiarisms arisen in programming exercises [7, 11]. A student may copy a Pascal or C program from another student. To detect this, the system pre-processes the programs to strip out all comments and extra blanks, and replaces all identifiers with numerical tokens. The most naive program copying by changing program indentation and variable names can be easily detected as the resultant programs will appear identical. To deal with possible changes in control constructs and positions of statements, condensed parse trees and their variants can be compared statistically [7]. Other schemes involve the calculation of program complexity metrics, such as Halstead's metrics [1, 14] and cyclomatic numbers [14]. Programs with complexity metrics differing by an amount less than prescribed thresholds will be flagged and a user will then determine if the programs are actually involved in plagiarisms.

The idea of copy detection servers for documents [3] is developed by extending existing program copy detection systems to address document plagiarisms. Original documents are registered at the server. Suspicious documents will be compared with the registered documents to check for plagiarism activities. Finally, a user will examine the potential plagiarized documents to determine if the documents truly violate copyright.

Detecting plagiarisms among documents are inherently more difficult than detecting plagiarized computer programs. Since computer programs are well-structured and the parse tree reflects the structure of a program clearly, a plagiarized computer program must ensure minimal changes to the parse tree of the original program in order to preserve the semantics of the program. On the other hand, a copied document can sustain a larger degree of changes in appearance, and yet still preserve the semantics of the original document. A parse tree which captures the layout of sections and paragraphs of a document [4, 13] is too coarse to be used because it is possible to have a completely different document but with a similar layout. A parse tree which captures the layout of sentences or words is, however, too fine to be useful since simply changing a sentence from active to passive form can disgust the parse tree and hence the copy detector. Methods that are based on a sentence-by-sentence comparison similar to the Unix diff command are often used for document comparison. They also suffer from similar problems in addition to the scalability problem mentioned in Section 1.

We observed that in reality, it will not serve any purpose to copy from a document describing an unrelated subject; comparisons between such documents could be eliminated. In our work, we incorporate information retrieval techniques to create a parse tree with additional information to represent the semantics of a document. This allows us to eliminate unnecessary comparisons based on the semantics of the documents rather than merely their syntactic appearances. Furthermore, by studying the semantics of the documents in additional to their syntax, we provide an extra level of security because users could no longer by-pass the detection mechanism by simply performing a minor modification to each sentence as the semantics of the documents would very likely remain unchanged. The parse trees for sections and paragraphs layout, though too coarse to be used directly, are useful when combined with information retrieval techniques.

3 THE ARCHITECTURE OF CHECK

The high level architecture of our copy detection prototype, CHECK, is depicted in Figure 1. As shown in the diagram, CHECK is composed of three main modules: *document registration*, *document comparison*, and *document parsing*. The document registration module registers an original document into a database server, which maintains the set of documents considered original. The document comparison module compares an input document against the registered documents to detect for any possible plagiarism. The document parsing module builds an internal indexing structure, called *structural characteristic* (SC) for each document to be used by the document registration and comparison modules. In CHECK, the Oracle database management system is adopted to maintain the SC of each document. The three modules of the system provide three basic functions: original document registration, document verification, and normal document registration.

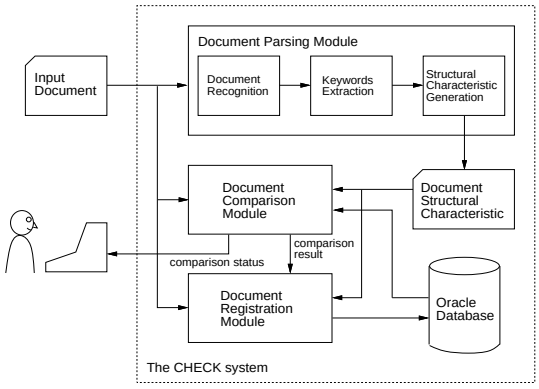

Figure 1: The architecture of CHECK.

1. *Original document registration*: A document which is believed to be original will be registered in the CHECK system. In this case, the document registration module would invoke the document parsing module to index the document into the Oracle database. Comparison with existing registered documents is not performed, since the procedure is relatively time consuming. Obviously, the user of this facility must be someone trustworthy. As such, this function must be privileged. All other documents should be registered using normal document registration function, as described below.

2. *Document verification*: A user can test a document, A, for plagiarism activities by invoking the document comparison module. The document comparison module will in turn invoke the document parsing module to build a structural characteristic (SC) for document A. The SC of A, SC_A, will be used to compare against the SCs of existing registered documents on a pair-wise basis (see Section 3.2). All suspicious documents found will be reported back to the user.

3. *Normal document registration*: This function basically calls the document verification function to verify if an input document is suspicious of plagiarisms. If it is not, the document is registered into the database via the original document registration function. If it is found to be suspicious, the result is reported to the user.

We describe the design and implementation of the three modules of the system in detail in the following subsections.

3.1 Document Parsing

The document parsing module is composed of the *document recognition*, *keyword extraction*, and *structural characteristic generation* submodules, operating in a pipelined fashion, as shown in Figure 1.

3.1.1 Document Recognition

The document recognition submodule converts a formatted document into a plain ASCII document. The current version of CHECK only recognizes LaTeX[9] documents, i.e., documents formatted using the LaTeXcommands. Other document recognizers such as those that convert DVI or troff documents into ASCII documents are being considered. When converting a LaTeXdocument into plain ASCII text, some formatting information including the hierarchical structure of a document and the specially formatted keywords are extracted. That is why we could not simply employ the Unix utility detex. The hierarchical structure of the document allows document comparison to be performed recursively at different levels of abstraction. This can eliminate some unnecessary comparisons by early termination of comparison (see Section 3.2). The specially formatted keywords, such as those words which are boldfaced, bear some important meanings within a document, thus providing some heuristic rules for keyword extraction (see Section 3.1.2).

Our LaTeXrecognizer performs a single pass on each document. During the pass, the recognizer creates a tree-like structure called *document tree*, which resembles the structure of the document. Each document could be viewed at multiple abstraction levels which include the document itself, its sections, subsections, subsubsections, and paragraphs, with paragraphs representing the lowest level of abstraction [13]. Figure 2*a* and Figure 2*b* depict a sample LaTeXdocument and its corresponding document tree. Here, the root of the document tree resembles the document and the children correspond to the sections of the document. Each section may be composed of subsections, with each subsection further composed of subsubsections and so on. Each leaf node of the tree resembles an individual paragraph of the document. Each node contains pointers to its children in the document tree as well as pointers to the first and last sentence of the corresponding organizational construct in the document file. For instance, in Figure 2*b*, the node for Section 3 contains pointers to its children nodes modeling Section 3.1 and Section 3.2. It also contains pointers to the first and last sentence of Section 3 in the document file.

As in [3], our LaTeX recognizer in the CHECK prototype performs several simplifications when parsing a

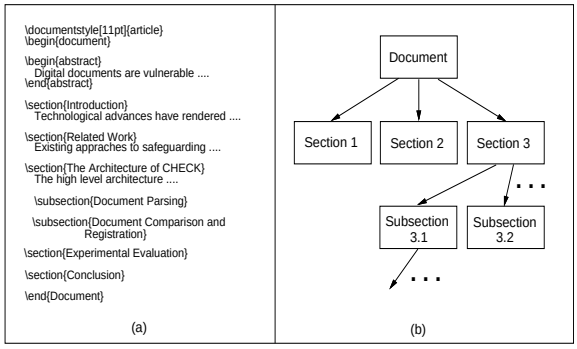

Figure 2: Sample document and corresponding document tree.

LaTeX document. Table, figure, and equation environments are ignored for simplicity. Each enumerated and bullet item is considered as a single paragraph. Finally, most of the LaTeX formatting commands are discarded except italics and boldface, whose formatted words are extracted as keywords (see Section 3.1.2).

For each input word in the document, the LaTeX recognizer converts it into a canonical form. This is performed with the aid of an intelligent lexicon, providing rules to convert all plurals to singulars, all verbs from any tense to their present infinitive form, and all suffixes such as adjectives and adverbs to their fundamental stems [5, 10]. For instance, a rule may specify that all words ending with the suffix ful should be converted to the fundamental stem by removing ful if the resultant word is in the lexicon. Another rule may describe the fact that all words ending with iful can be converted by replacing iful with y if the resultant word is a legal word. This can greatly reduce the size of the lexicon and hence improve the comparison performance.

3.1.2 Keyword Extraction

The keyword extraction module employs information retrieval techniques to identify keywords that would best describe the semantics of a document. In general, each word can be categorized into *open-class* or *close-class*. Open-class words including nouns, verbs, adjectives, and adverbs are usually meaning bearing, and are potential candidates for keyword extraction. In contrast, close-class words including pronouns, prepositions, conjunctions, and interjection are usually not meaning bearing and are not considered as keywords. Hence, the document recognition module parses each word in an input document and converts it into its canonical form. The keyword extraction module determines if it belongs to open-class or close-class by consulting the intelligent lexicon again.

Not all identified open-class words are qualified as keywords for a document because different words might bear different semantics with respect to the document. For each identified open-class word, a, we keep track of its number of occurrences, $|a|$, within the document. The mean occurrences, μ, of all open-class words and the corresponding standard deviation, σ, are computed. Each open-class word, a, will be qualified as a descriptive keyword for the document if its number of occurrences, $|a|$, falls within the *threshold interval* $[\mu-\alpha\sigma,\ \mu+\beta\sigma]$ where α and β are two adjustable user-defined parameters. Notice that since close-class words have been eliminated, all words within the range of the highest frequency should be most descriptive with respect to the content of the document. This is in contrast with traditional information retrieval approach in which close-class words are also considered in frequency distribution analysis. Words within the range of highest frequency will be close-class words in traditional approaches. For example, the word "computer" can be considered as a close-class word as it appears almost everywhere in a computer science document and should not serve as a keyword for the purpose of information retrieval. It can, however, serve as a keyword to distinguish the document type. In CHECK, the value of β can be set to a very large value if it is desirable to treat these words as keywords.

In addition to frequency distribution analysis, we also employ some heuristics in keyword extraction. We notice that in general, the italic and boldface formatted words bear some special meanings within a document, such as a newly defined term. The keyword extraction module, therefore, explicitly looks at this kind of formatting commands, i.e., {\em} and {\bf}. All such formatted words would be regarded as keywords even if their number of occurrences fall beyond the threshold interval. Other formatting commands such as {\sl} might be useful as well, but are highly related to the writing habit of individual authors. We are investigating into more heuristic rules of this sort.

3.1.3 Structural Characteristic Generation

The structural characteristic (SC) of a document, A, is created by merging its document tree with its set of keywords, K_A. For each node, n_i, in the document tree, we assign a set of keywords for it as follows:

1. If n_i is the root of the document tree, resembling document A, the set of keywords for n_i will be K_A since K_A contains the keywords that best describe the document. Each keyword of K_A, a, will be associated with a weight,

$$w_{K_A,a} = \frac{occurrence\ of\ a\ in\ A}{\sum_{x \in K_A}\{occurrence\ of\ x\ in\ A\}},$$

   indicating its degree of importance with respect to the document.

2. If n_i resembles section i of document A, the keywords, K_i, assigned for n_i will contain only those keywords in K_A that appear in section i of the document. Again, each keyword a of K_i will be associated with a weight,

$$w_{K_i,a} = \frac{occurrence\ of\ a\ in\ section\ i}{\sum_{x \in K_i}\{occurrence\ of\ x\ in\ section\ i\}},$$

   indicating its degree of importance with respect to section i of A.

3. If n_i resembles a subsection, subsubsection, or paragraph of document A, the keywords and their weights assigned for n_i is similar to step 2 except that the occurrence of a keyword in the respective organizational construct will be used.

This multi-level SC of a document provides a multi-resolution description for the document; the highest level SC node (the root node) provides a high-level semantic description for the document while lowest level SC nodes (the leaf nodes) provide the specific description for individual paragraphs. The keywords and their associated weights for each node are used by the document comparison module and will be described next.

3.2 Document Comparison and Registration

The document comparison module compares the SC of an input document, which is generated by the document parsing module, with the SC of each registered document in the database. Recall that the SC is simply a document tree with weighted keywords assigned to each node of the tree. The similarity between the set of keywords for a node of an SC and that of another SC indicates the degree of similarity in content between the modeled organizational constructs. An advantage of comparing the SCs instead of the documents is that unnecessary comparisons can be filtered at any level of SCs recursively, in a depth-first search manner.

The document registration module registers an input document to the system and stores both the document and its SC in the database for future comparison. A user can either input a document to the system and indicate that it is an original document. In this case, the system will simply call the module to register the document. Alternatively, a user can request the system to check the document before registering it. In this case, the system will call the document comparison module to check for the validity of the document. If no similarity is found between the input document and any of the existing registered documents, the system will call the document registration module to register the input document.

To discuss how the SCs of two documents are compared, let us denote the SC of an input document A as SC_A and the SC of a registered document B as SC_B. The comparison between SC_A and SC_B is carried out as follows:

1. Starting from the root of the two SCs (level 0), the similarity between the sets of keywords, and hence the similarity between the two root nodes, is determined. Let us denote the set of keywords for document A as a vector, $V_A = [a_{A,1}, a_{A,2}, \cdots, a_{A,|V_A|}]'$, and their associated weights as another vector, $W_A = [w_{A,1}, w_{A,2}, \cdots, w_{A,|V_A|}]'$, where $a_{A,i}$ and $w_{A,i}$ represent the i^{th} keyword and its weight in document A respectively. Similarly, we denote the set of keywords and their associated weights in document B as vectors $V_B = [a_{B,1}, a_{B,2}, \cdots, a_{B,|V_B|}]'$ and $W_B = [w_{B,1}, w_{B,2}, \cdots, w_{B,|V_B|}]'$ respectively. The similarity between V_A and V_B could be determined in two steps: *normalization* and *dot product*.

   Normalization: Since V_A and V_B might contain different number of elements, the two vectors must be normalized to the same length. Furthermore, since the two keyword vectors have symbolic terms as members, the symbolic vectors must also be normalized into numerical vectors for comparison. A reference vector, R, is first generated as a union of the elements in both V_A and V_B, i.e., $R = V_A \cup V_B = [a_{R,1}, a_{R,2}, \cdots, a_{R,|R|}]'$ and $|R| \leq |V_A| + |V_B|$. Let us denote the corresponding normalized vector of V_A as $\chi_A = [x_{A,1}, x_{A,2}, \cdots, x_{A,|R|}]'$ where $x_{A,i}$ is a value in the interval [0,1] representing the weight of the i^{th} keyword of R, $a_{R,i}$, in the context of document A. Formally, $x_{A,i}$ is defined as follows:

$$x_{A,i} = \begin{cases} 0 & \text{if } a_{R,i} \notin V_A \\ w_{A,j} & \text{if } a_{R,i} = a_{A,j} \in V_A \end{cases}$$

   The normalized vector of V_B, χ_B, can be computed in a similar manner.

   Dot product: With the two normalized weight vectors, χ_A and χ_B, the similarity between V_A and V_B, $S(V_A, V_B)$, could be simply determined by computing the dot product of χ_A and χ_B after they are converted into unit vectors [5]:

$$S(V_A,\ V_B) = \frac{\sum_{i=1}^{|R|} x_{A,i} \cdot x_{B,i}}{\sqrt{\sum_{i=1}^{|R|} x_{A,i}^2 \cdot \sum_{i=1}^{|R|} x_{B,i}^2}}$$

   Intuitively, the dot product between two unit numerical vectors represents the cosine of the angle between the two vectors in a two dimensional plane. The similarity between two nodes is thus characterized by determining the cosine of the angle between the two keyword vectors. A small angle between the two vectors will result in a high $S(V_A, V_B)$ value, which indicates a high similarity between the two SC root nodes.

2. If $S(V_A,\ V_B) < \varepsilon_0$, the *similarity threshold* for the root level, documents A and B are regarded as describing different subjects. In this case, there is no point to further compare the two documents. The document comparison module will therefore proceed to compare document A with other registered documents.

3. On the other hand, if $S(V_A,\ V_B) \geq \varepsilon_0$, documents A and B are considered to be describing similar subjects. In this case, the comparison between the SCs of documents A and B will proceed to the next level. For each level $l > 0$, comparison between two nodes of the two SCs is similar to step 1. Whether the comparison will proceed to the children of the two nodes depends on the similarity threshold at level l, ε_l.

4. This procedure is repeated for each level of the two SCs of documents A and B until the leaf nodes of the SCs are reached, i.e., until it is determined that two paragraphs of the two documents are highly similar, or until similarity between any two nodes

at some level all falls below the similarity threshold at that level. In the former case, we need to explicitly investigate the two paragraphs in a sentence-by-sentence fashion to determine if the two paragraphs have substantive overlapping. The approaches proposed in [3] could then be employed in this final step.

To illustrate the comparison process, consider the two SCs of two documents A and B as shown in Figure 3. In the diagram, the dashed arrows represent the path for a typical comparison process. The diagram illustrates that the root nodes of the two SCs are determined to be similar. The comparison process then proceeds to compare the children of the root nodes, i.e., at level 1. The three nodes of SC_A are compared with the three nodes of SC_B in a pairwise manner. If the node modeling Section 2 of document A is determined to be similar to the node modeling Section 3 of document B, the comparison process will proceed to compare the children of Section 2 with the children of Section 3. If the node modeling Section 2.2 is determined to be similar to the nodes modeling Section 3.1 and Section 3.2, the children of Section 2.2 will be compared with the children of Section 3.1 and Section 3.2. This comparison process will continue to the leaf nodes or stop at some level if the similarity among the nodes at that level all falls below the similarity threshold.

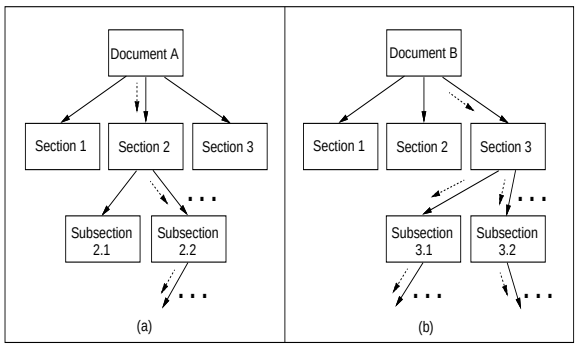

Figure 3: Document comparison process.

Notice that we employ a search technique in which the comparison between two nodes will be terminated once we determine that the organizational constructs modeled by the two nodes describe different subjects. This allows us to prune the search space and eliminate the comparison between unrelated branches of the SCs. The similarity threshold value for each level of the SC, ε_l, governs to what extent the two organizational constructs are considered similar.

4 EXPERIMENTAL EVALUATION

We have conducted some preliminary experiments to study the effectiveness of the CHECK prototype. Due to space limitations, only four sets of experiments are presented here. The purpose of these experiments is to study the effectiveness of SC in identifying plagiarism activities. For the purpose of our experiments, we define a plagiarized paragraph from an input document as one that is closely similar in semantics to a paragraph in a registered document, i.e., the dot product between the normalized keyword vector of an input paragraph and that of a registered paragraph exceeds the similarity threshold for the leaf nodes of the two corresponding SCs involved. Notice that, in practice, two highly similar paragraphs need to be further compared in a per-sentence basis to determine if any plagiarism activity is actually involved. Our experiments only study how many paragraphs will be involved in per-sentence comparisons. To evaluate the effectiveness of a comparison between two documents, we have modified traditional *recall* and *precision* metrics for information retrieval [5] as follows:

Recall metric: The recall metric measures the ability to retrieve a piece of information from candidate information. In our context, it is defined as the percentage of paragraphs identified as plagiarized with respect to the actual total number of plagiarized paragraphs between two documents, i.e.,

$$\text{Recall} = \frac{number\ of\ plagiarized\ paragraphs\ identified}{total\ number\ of\ plagiarized\ paragraphs}$$

Precision metric: The precision metric represents the ability to retrieve a piece of information correctly. Here, it is defined as the percentage of plagiarized paragraphs identified with respect to the total number of identified paragraphs, i.e.,

$$\text{Precision} = \frac{number\ of\ plagiarized\ paragraphs\ identified}{total\ number\ of\ paragraphs\ identified}$$

Document ID	Subject	Plagiarism Activities
A	I	No Copying
B	II	No Copying
C	III	No Copying
D	I	Copy from A
D'	I	Copy from B
D"	I	Copy from C
E	II	Copy from B
E'	II	Copy from C
E"	II	Copy from A
F	III	Copy from C
F'	III	Copy from A
F"	III	Copy from B
G	I	No Copying
H	II	No Copying
I	III	No Copying

Table 1: Test documents distribution

Our set of testing documents consists of 15 technical papers and are distributed as in Table 1, where I, II, and III denote three different subjects. In other words, documents A, D, D', D", and G are technical papers describing subject I; B, E, E', E", and H are papers describing subject II; C, F, F', F", and I are papers describing subject III. Documents D, D', and D" are in fact, the same technical paper except that they have different plagiarism activities, copying from documents A, B, and C respectively. Similarly, documents E, E', and E" are the same document copying from documents B, C,

and A respectively while documents F, F', and F" are the same document copying from documents C, A, and B respectively. Documents G, H, and I describe similar subject as documents A, B, and C respectively, but with no plagiarism activity.

The documents contain an average of about 8000 words and 500 sentences in length, structured into 50 paragraphs. When copying from a document, six paragraphs are randomly picked from the copied document. Six paragraphs in the copying document will be replaced. If the copied and copying documents describe similar subject, the replaced paragraphs are chosen manually to ensure that the replaced and replacing paragraphs have similar semantics. If the documents describe unrelated subjects, we try the best to ensure that the replacing and replaced paragraphs are from the same organizational construct. For instance, if a replacing paragraph is picked from a section called Introduction, the replaced paragraph will be picked from the same section as well. In case it is not possible to pick the replaced paragraphs from the same organizational construct as the replacing paragraphs, we simply pick the replaced paragraphs randomly.

When creating SCs for the testing documents, the parameters α and β of the threshold interval during keyword extraction are both set to 1. We have not experimented the effect of α and β on the effectiveness of our system and is left for future work. The similarity threshold values for all levels l, i.e., ε_l, are set to 0.5, unless otherwise specified.

4.1 Experiment #1

In the first experiment, we would like to test if the CHECK prototype is able to identify identical documents. We have conducted three independent experiments, comparing documents A, B, and C against itself in each experiment. Each document is first submitted for registration, and then, re-submitted for comparison. The average value of recall for the three experiments is 100%, i.e., CHECK is able to detect that all paragraphs are copied. The average value of precision for the three experiments is approximately 90%. In other words, for each experiment, CHECK reports approximately 5 plagiarized paragraphs that in fact do not violate plagiarism. The primary reason for this anomaly is that authors tend to reuse sentences at different places within a document.

4.2 Experiment #2

In the second experiment, we would like to study the behavior of CHECK on documents describing similar subject. We compare different original documents describing similar subject: G against A, H against B, and I against C in a similar fashion as in Experiment #1. For instance, document A is submitted for registration and document G is submitted for comparison with document A. The purpose of this experiment is to study if CHECK would be able to tell that the documents are different. The recalls from three comparisons are all equal to 0. Hence, CHECK is able to detect that the documents are all original. We have instrumented the comparison process to determine at which level of the SCs the comparison process stops. We found that the comparison process stops at level one. This shows that CHECK is able to determine that the two documents are describing similar subject as the comparison process passes through the root of the two SCs, i.e., level 0, but CHECK is also discriminative enough to distinguish that they are in fact two different documents.

4.3 Experiment #3

The first two experiments test whether the CHECK prototype satisfies the necessary system requirement. It should be able to detect an identical copy, which is tested in Experiment #1, and to detect a non-copy, which is tested in Experiment #2. The purpose of our third experiment is to study if CHECK could detect actual copying activities which is the sufficient system requirement. In this experiment, we compare documents D against A, E against B, and F against C.

The average precision for the three comparisons is 100%. Hence, all detected paragraphs are indeed plagiarized paragraphs. However, the average recall ranges approximately from 50% to 90% depending on the value of the similarity threshold for each level, which ranges from 0.7 down to 0.3. This behavior is primarily due to the small degree of overlapping between the two organizational constructs of the documents with only one to two paragraphs overlapped. In addition, the higher the similarity threshold, the higher the similarity measurement between two SC nodes is required in order for the comparison process to proceed to the next level. This results in a lower recall. As the similarity threshold decreases, a higher recall can be achieved.

4.4 Experiment #4

Our last experiment reported is to study the behavior of CHECK when testing plagiarized documents describing unrelated subjects. We compare documents E" and F' against A, documents F" and D' against B, and documents D" and E' against C.

The recall value for the six comparisons are all equal to 0. In other words, CHECK is not able to detect any plagiarized document if the copying documents describe different subjects from the copied documents. This behavior is, in fact, very controversial. One might argue that any plagiarism activity should be detected regardless of the semantic meaning of the documents. However, we believe that it usually serves no purpose nor benefit to copy from an unrelated document.

In fact, the CHECK system could operate in collaboration with the copy detection mechanism proposed in [3]. If the goal is to detect any plagiarism activity regardless of the content of the documents, the exhaustive copy detection mechanism proposed in [3] should be employed. Otherwise, the CHECK system should be employed.

4.5 Security Measurement

Any copy detection system is vulnerable if a malicious document could by-pass the detection mechanism easily. To understand how secure our detection mechanism is, we have conducted several simple experiments. Due to space limitation, we only describe a representative here.

We randomly pick several words from each sentence of document A and remove the picked words from the sentence. The number of words removed is determined randomly, ranging from 1 to the number of words in the sentence. The resultant document, A^*, is compared with the original document A. The same experiment is repeated with documents B and C. We repeat each experiment 30 times and the average recall and precision of the 90 experiments are measured. With a similarity threshold ranging from 0.7 to 0.4, the results of the experiments are similar to those in Experiment #1. In other words, CHECK is still able to detect that the two documents have a large degree of overlapping since the semantic meaning reflected by the structures of the documents, sections, subsections, subsubsections, and even paragraphs remain largely the same.

5 CONCLUSION

In this paper, we have described a plagiarism detection system that allows plagiarized documents to be detected. Several experiments have also been described to illustrate the feasibility of our mechanism. Although the results of our experiments are specific with respect to our testing set of documents, we have illustrated the general behavior of our CHECK prototype.

One behavior we have not discussed in this paper is the complexity of the comparison process. We are currently studying the number of SC nodes the comparison process needs to evaluate before CHECK could decide if two documents have any substantive overlapping. This has a direct implication on the amount of time each comparison needs. We would also like to compare the time required for our mechanism with the time requirement for existing mechanisms [3].

To evaluate the CHECK prototype in a more rigorous manner, we are currently collecting a large set of technical documents, covering different areas such as computer science and economics. This will provide a fairer evaluation on the CHECK detection system. We are also testing the CHECK prototype against a taxonomy of different copying activities such as copying from several related documents or merging several organizational constructs into one. The behavior of CHECK under different copying activities would be reported in the future.

Finally, we would also like to enhance the document parsing module to recognize documents in other common formats, such as DVI and troff documents as well.

ACKNOWLEDGEMENTS

We would like to thank Ken Lee and Leon Sum for their assistance in implementing the CHECK prototype. This research is supported in part by the Hong Kong Polytechnic University Central Research Grant numbers 353/066 and 340/115.

References

[1] H.L. Berghel and D.L. Sallach. Measurements of Program Similarity in Identical Task Environments. *SIGPLAN Notices*, 19(8):65–76, August 1984.

[2] 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, 1994.

[3] S. Brin, J. Davis, and H. Garcia-Molina. Copy Detection Mechanisms for Digital Documents. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 398–409, 1995.

[4] M. D. Dunlop and C. J. Rijsbergen. Hypermedia and Free Text Retrieval. *Information Processing & Management*, 29(3):287–298, 1993.

[5] W. Frakes and R. Baeza-Yates. *Information Retrieval: Data Structures & Algorithms*. Prentice-Hall, 1992.

[6] G. N. Griswold. A Method for Protecting Copyright on Networks. In *Proceedings of Joint Harvard MIT Workshop on Technology Strategies for Protecting Intellectual Property in the Networked Multimedia Environment*, 1993.

[7] H. T. Jankowitz. Detecting Plagiarism in Student Pascal Programs. *The Computer Journal*, **31**(1):1–8, 1988.

[8] JPEG. JPEG Digital Compression and Coding of Continuous-Tone Still Images. Technical Report ISO 10918, ISO, Draft, 1991.

[9] L Lamport. LaTeX*: A Document Preparation System*. Addison Wesley, 1986.

[10] Y. Maarek, D. Berry, and G. Kaiser. An Information Retrieval Approach for Automatically Constructing Software Libraries. *IEEE Transactions on Software Engineering*, pages 800–811, 1991.

[11] A. Parker and J. O. Hamblen. Computer Algorithms for Plagiarism Detection. *IEEE Transactions on Education*, **32**(2):94–99, May 1989.

[12] G. J. Popek and C. S. Kline. Encryption and Secure Computer Networks. *ACM Computing Surveys*, **11**(4):331–356, 1979.

[13] R. Rada, W. Wang, and A. Birchall. Retrieval Hierarchies in Hypertext. *Information Processing & Management*, 29(3):359–371, 1993.

[14] Martin L. Shooman. *Software Enginnering*. McGraw Hill, 1985.