International Journal of Security and Its Applications
Vol. 10, No. 12 (2016), pp.289-298
http://dx.doi.org/10.14257/ijsia.2016.10.12.23

Copy Detection Technique with Enhanced Efficiency by using Substring Comparison Algorithm

U. Nanaji[1], N. Thirupathi Rao[1], Ch. Raj Kumar[1], Debnath Bhattacharyya[1], Hye-jin Kim[2]

[1]Department of Computer Science and Engineering,
Vignan's Institute of Information technology
Visakhapatnam-530049, AP, India
{nanajistiet,nakkathiru,debnathb}@gmail.com
[2]Sungshin Women's University,
2, Bomun-ro 34da-gil,
Seongbuk-gu, Seoul, Korea
hyejinaa@daum.net
(Corresponding Author)

Abstract

As the information was being overloaded from time to time, a significant technique should be accomplished by several search engines that exists will use from the databases to eradicate the replicas of data which was available in articles and to present the outcomes of the search to users in terms of percentage of the amount of the data copied from the original file. Plagiarism is the process of bestowing one's creative ideas as our individual conception. It is does not meant that anybody cannot use other's considerations or workings, anybody can use their data by giving the actual credit the original users by stating their names in literature or in references and also by giving special note on acknowledgements. Copying the content of the others work is mainly considered to be a big crime in terms of research and in terms of the owning the idea of the others. The main idea of this work is to identify the extent of the data that was being copied or to identify the amount of data that was being copied from other peoples work or their own credited work. Performance of plagiarizing a document is not limited to word files or pdf files, it is possible to plagiarize even images and other files too. Hence, we were focused on finding the plagiarism in word files which includes academics and research articles.

Keywords: *Plagiarism, duplicated copy, word files, pdf files*

1. Introduction

Plagiarism is the most worried problem for any researcher or any other writers writing or presenting a paper or an research article in a journal or in an conference in any field of technology or in medical or in some other arrears of research. This problem of plagiarism is accomplishing more and shoddier due to the accessibility of original documents to each and every user in the internet or in web in the form of a word file or in a pdf files [1].

Plagiarism can be done or can be observed in two modes of usage. They are,

1. It is observed in programming languages
2. It is observed in natural languages

The different techniques that are available to identify the plagiarism either in programming languages or in natural languages are different. The key features or terms like number of lines used or number of lines copied or number of lines used for representation, usage of variables in representation or usage of variables in declaration

ISSN: 1738-9976 IJSIA

etc, and the plagiarism in natural languages can be used or detected by using several features like based on text etc.

2. Proposed Work

2.1. Document Registration Module

The document registration component registers documents in the database. It is the least complex of the major components; most of the work is performed by library functions that interface with the database component. Furthermore, since the design of the database component allows documents to be added dynamically to existing databases, the registration subsystem can either add documents to an existing database without needing to invoke separate algorithms. Given the design of the database component described, document registration is straightforward. We described the process of registering a list of documents with the system. A list L of documents to be registered and a database name, either existing or new, are passed in to the algorithm. Less importantly, the database components, hash function, and sentence-splitter, are also passed into the algorithm.

2.2. Copy Detection Module

The copy detection component finds the overlap that occurs between a test document and a database of registered documents. In a nutshell, it finds the overlap by comparing the hash values of the substrings in the documents. Since hash collisions are extremely rare, when a match occurs we assume that a substring in the test document is also found in a registered document. In this section, we discuss the creation and purpose of the three core data structures used by the copy detection component: the hashlist, the matchlist, and the file_index and discussion of the behavior and operation of this component, which is central of the system. The hashlist is a simple array containing the hash values of an input document. The input document is first split into sentences. Each sentence is then hashed and appended to the hashlist. As a result, the hashlist is structured so that hashlist[i] holds the hash value for the ith sentence.

2.3. Collision Analysis Module

The collision analysis component analyzes the collision behavior of a hash function. The database subsystem, capable of storing several million sentences, serves as a tested for hash function analysis. Furthermore, the component-based design of the entire system allows the collision analysis component to perform its function without interfering with the other components. As with the copy detection component, the collision analysis component makes extensive use of the query capabilities of the database component. The collision analysis component must separate actual hash collisions from potential hash collisions. Potential collisions occur when two sentences hash to the same hash value, but an actual collision occurs only if the two substrings are distinct. If the substrings happen to be identical, no collision occurs.

In other words, suppose that sentences S1 and S2 hash to the same value. If S1, S2, then the sentences collide. There is no collision if S1 = S2. Separating the actual collisions from the pool of potential collisions requires that the sentences corresponding to the hash values be compared. Since the database contains only the hash values and other associated information, the registered documents must be accessible in order to determine whether S1 and S2, as, above, are identical or not. For this reason, the collision analysis component requires access to the registered documents.

Stage 1: The document should be preprocessed at initial stage

The first step is to preprocess the data by reading the documents and they were compared by fragmenting them into several objects like objects with sentences and the present object comprises of a list of words that were identified from the original text. Once the sentence is fragmented to a list of words, it uses two filters,

1. The first filter is used to convert the words which were wrote in lower case can be converted such that to reduce the time for comparing the same type of documents.
2. The second filter is used to eliminate the white spaces if any exists between the documents which were to be verified or checked.

Stage 2: In the second stage, the process of comparing the documents should be performed here

In the second stage, the documents that were obtained from the above two filters are compared by verifying the substring from one document to the other document thoroughly. The result was computed in terms of similarity score of which is based on the metrics like word count. The score that had observed from the above procedure is the regular resemblance among the substrings, which was calculated as a function of words, and the lengths of the substrings which were going to be calculated and should be compared.

In order to overcome the drawbacks faced in the existing copy detection system, here we are proposing Substring Comparison algorithm.

2.4. Substring Comparison Algorithm

The documents are compared after being converted to all lowercase letters and stripped of all whitespace. Substring Comparison looks for identical substrings shared between two documents. This algorithm compares doc1 and doc2 by looking at every substring s1 of length "tolerance=1" in doc1, and seeing if s1 exists in doc2. By fixing the tolerance value to 1, we can compare each character in doc1 with every character in doc2. And by increasing the tolerance value to more than 1 we can compare the two documents at the sentence level.

Substring Comparison Algorithm:

algorithm character table formation for
comparison :
input:

The word to be analyzed can be taken as an array of characters W
The table which has to be filled can be taken as array of integers T

output:

Nothing (but during operation, it populates the table)

3. Frame Work Design for Proposed Approach

The following is the frame work we used to find the Substring Comparison Model document similarity detection method,

Stage 1: In the first stage, the similarity between the two docs, query document d_q and document collection D or the pair of documents (d_q; D_x) was calculated.

Stage 2: In the second stage, the set of candidate documents from document collections was collected.

Stage 3: In the third stage, the indexing of both the documents which we had collected from above two stages was performed and the splitting of the document was performed before doing the indexing operation in the documents.

Stage 4: This is the final stage of the frame work which we had proposed, developed and analyzed to find the similarity. Here the individual terms and their frequencies of test and listed documents for comparison were considered to figure out the resemblance among the words of both the documents.

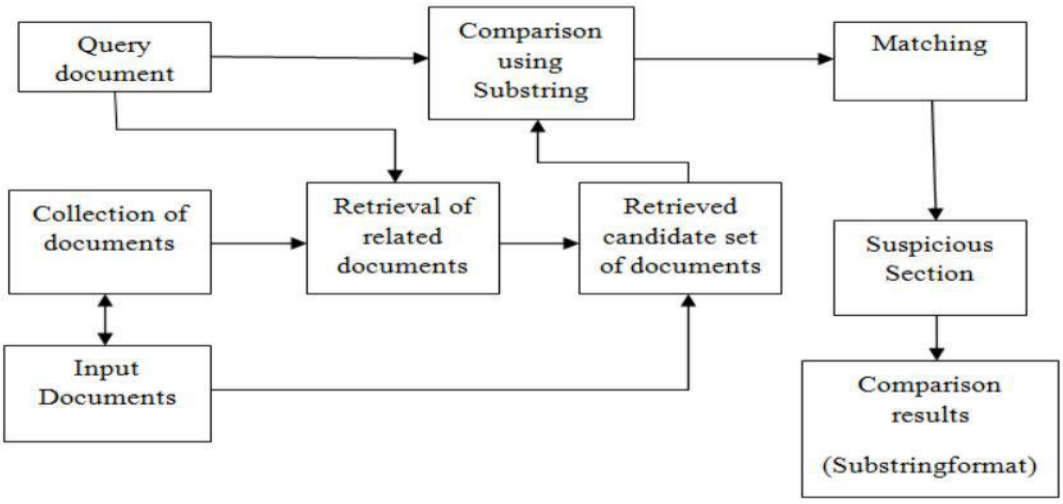


Figure 3. Frame Work for Substring Comparison Model

Finally we are with the suspicious sections (substring) (s_q; s_x). Where (s_q; s_x):s_q 2 d_q; s_x 2 d_x; d_x 2 D

For the implementation purpose we have omitted the step of retrieving candidate set documents by considering the assumption of taking related documents as input for the plagiarism detection system.

4. Result Analysis

4.1. High-level Architecture Description

Figure 3 shows a high-level depiction of the copy detection system. In some sense, this system is just a collection of interacting components. In this, we describe the system according to this component-based paradigm. In addition, when the context is clear, we use the terms “system”, “subsystem”, and “component” interchangeably. Creating a component-based system has three distinct advantages. First, without altering the operation or the functionality of the other components, new components can be added or removed with minimal effort. This component-based approach allows us to easily “plug-in” new hash functions or new sentence boundary recognizers for experimentation.

The query document component is used to store or collect the query for which the data has to be verified or identified for the case of plagiarism. After the query had been generated, the data has to be found from the database that whether the related items or the items related to the query has to be found from the existing databases and also from the other source of databases. The query will be then passed to the substring algorithm to identify the items that were being stored or identified in the databases at various places. If the match was found from the database with the selected questions or the selected data, this data was reported to the suspicion section. Then the suspicious section will display

the resulted data to the end user, if the user requests the suspicious data, then the data will be displayed to the end user in the form of both the existing data and the data that was found to be copied form the other documents.

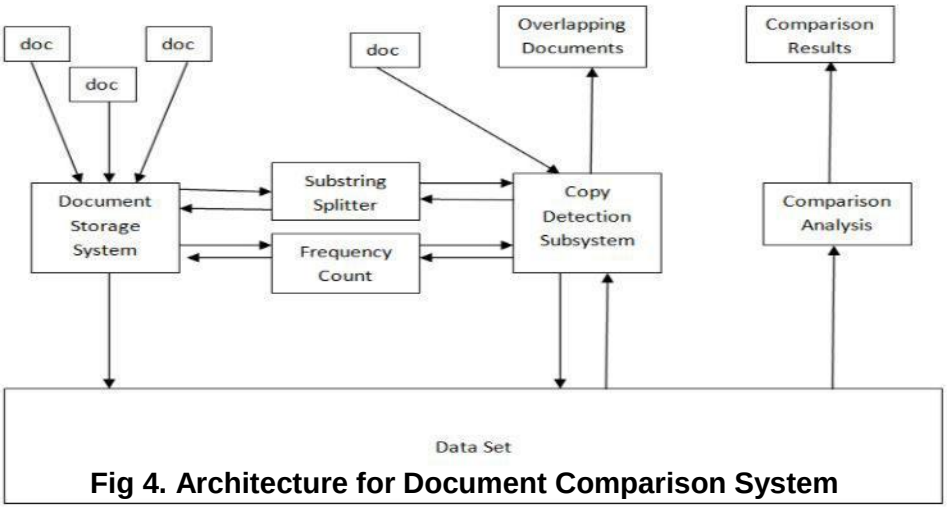


Fig 4. Architecture for Document Comparison System

The architecture of the document comparison system was shown from the above figure 4. The present comparison architecture gives the detailed explanation of the working style of this comparison system. The system comprises of the document storage system which stores the several documents that were being placed in the internet at various servers or at various databases which can be used for matching the documents to identify the similarity between the documents. The data or the documents that were being collected from the storage system are then passed to the substring splitter and to the frequency count. The substring splitter splits the given document or the set of lines of content in to various numbers of sub sentences or to the sub splits for easy comparison process. The duty of the frequency count unit is to count the rate at which the documents are being split in to sub sentences.

Then the documents of both the similar documents and the actual document which needs to be identified were fed to the copy detection system which was used to detect the amount of data that was being copied from the other sentences to the sentences which were written in the same area or in the same topic of the target sentence to be verified. The documents which were overlapped or the documents which were identified as the copied documents from the original document to the given document as input document should be placed in the unit of overlapping documents unit. All the documents which were matched with respect to the content in terms of sentences were kept here in this unit without considering the format of the data document of which that might be in either word document or in pdf document model.

After processing all the data from the units of these, the data and the results were being sent to the data set where all the details regarding the data that was searched for comparison was stored. When the user requests the results that were obtained from the data set about the matching of the documents with the previous data sets, the results were thoroughly analyzed. The analyzed results will be placed by highlighting the documents with different colors. The highlighted color data will be shown with the support of the results and the documents that were being copied with the proper links such that to verify the links if the user needs to verify the results. The detailed results and the result copy were made available to the end user in the results unit of the system.

5. Database Design

The database system is composed of the six tables depicted in Figure 4. These six tables are as follows:

- the hash table (Table 1),
- the character-based offset table (Table 2),
- the character-based mapping table (Table 3),
- the sentence-based mapping table (Table 4),
- the names of registered documents (Table 5), and
- a table containing internally-used data (Table 6).

From a functional perspective, the database system stores registered documents. During registration, a document is split into its component sentences, each sentence is hashed, and the hash values are placed in the database. However, as the tables in Figure 5.2 indicate, a hash value from a document is not grouped according to the document from which it comes. Instead, the database assigns to each hash value a sentence-based offset and a character-based offset reflecting the order in which the hash value was added. These offsets are uniquely assigned to each hash value in the database; they are not re-set or otherwise modified when a new document is registered.

So, for example, when the 250,000th sentence is to be added to the database, its corresponding hash value is coupled with sentence-offset 250,000. Similarly, the 250,000th sentence is assigned the character-based offset corresponding to the total number of characters in each of the first 250,000 sentences, plus one. The character-based and sentence-based offsets enable a design that distributes data throughout the tables. In table 1, the hash table, the <hash value, global sentence offset> pairs from the registered documents are intermixed. Tables 2, 3, and 4 contain data sufficient to resolve any intermixed pair in table 1 into its sentence-based and character-based offset relative to the registered document from which the pair originates.

Unlike typical relational databases, the tables are structured as files of contiguous, fixed-width records, and each table in the database can simply be thought of as an array of records stored on disk. Existing database records are never modified; the database is read-only except for the registration process, which adds records to the database and modifies table 6. With the exception of table 1, a new document is registered simply by appending its sentences' hash values and other data to the existing tables.

5. Copy Detection

The major work of this unit in the detection system or the copy detection system was to identify the amount of data that was being copied from a document or some other source that might be either in a word document or in a pdf document. In the present application, the detection of the data that was being copied was identified only in terms of the sentences. The other types of data the headings and the images with the data was not identified by using this model of system. These model best suited for the identification of the amount of data that was being copied from the other sources of documents that were in the format of word document format or in the pdf document format. The hash list is a simple array containing the hash values of an input document. Its creation is outlined in the below Algorithm. The input document is first split into sentences. Each sentence is then hashed and appended to the hashlist. As a result, the hashlist is structured so that hashlist[i] holds the hash value for the ith sentence.


```
Void BuildMatchlist(hashlist: ARRAY;
                    Db: Database);

BEGIN
4.
    FOR EACH hash value H IN the hashlist DO
5.
    BEGIN
6.
        id := Binsearch(H, Db, Table1);
        IF (id = <NOT_FOUND>)
        THEN
7.
            RETURN;

        Ri := GetRecord(Table1, id);

        (Ri-j..Ri+k):= sequence of contiguous records such that the hash
        values of each record are identical;
8.
        FOR EACH R IN (Ri-j..Ri+k) DO

        BEGIN

            Create a match record M from R;

            //Append M to the list stored at index R.H of the matchlist, //where R.H is
            the hash value stored in record R;
9.
            Append(Matchlist[R.H],
            M); END;
10.
```

Table 1. Hash table - Entries are Sorted according to Hash Value

<hash value, global sentence offset>
.
.
.
hv_a, gso_i
hv_a, gso_j
hv_b, gso_k
hv_c, gso_l
.
.

In the above table 1, the entries in the hash table are sorted according to the hash value which was required in the comparison model.

From the below table 2, the character based offset table was observed and the character based offset of second sentence, the third sentence and so on for the ith sentence.

Table 2. Character–based Offset Table

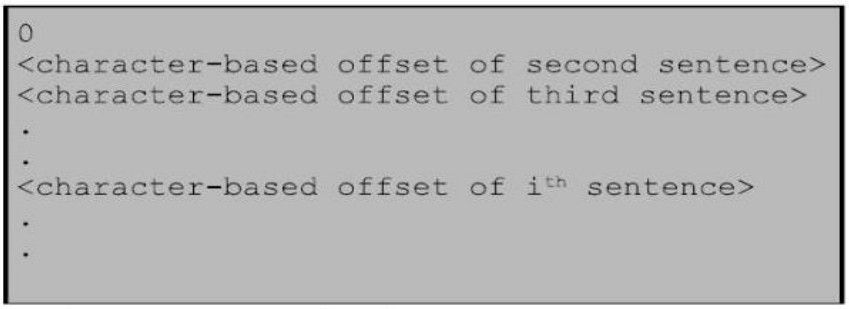
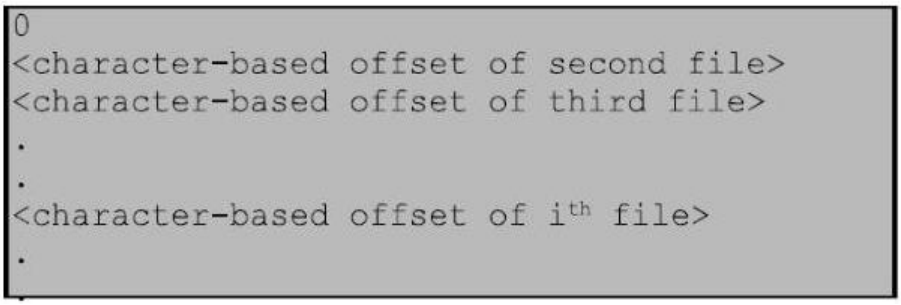
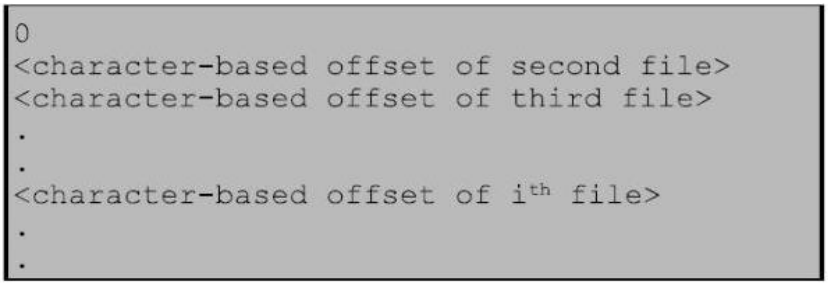


Table 3. Map: Global Character Offset →file-based Character Offset



From the above table 3, the global character offset values to the file based character offset values were declared. The character based offset value of second file, character based offset value of third file and the character based offset value of ith files were declared consecutively.

Table 4. Map: Global Sentence Offset → file-based Sentence Offset



From the above table 4, the global sentence offset value to the file based sentence offset was represented. Here the character based offset value of the second file, the character based offset value of the third variable and the character based offset value of the ith file were declared consecutively.

Table 5. Names of registered document

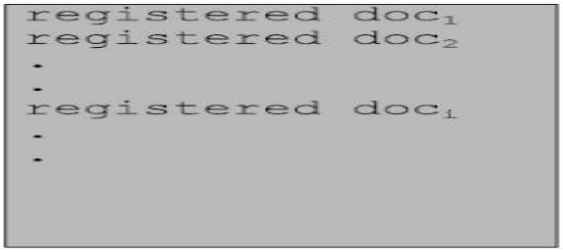


Table 6. Internal Data Needed to Dynamically add New Document

<next character offset> <next sentence offset>

From the table 5, the documents that were already registered were displayed. The diagram represents the registered document 1, registered document 2 and the registered document of ith number. Similarly, the table 6 represents the internal data that was required to add a new document to the searching mechanism dynamically. The data that was represented in the above figure was the next character offset value and the next sentence offset value that has to be represented in the search of the documents.

6. Experimental Comparison

Figure 5 shows the comparison between sentence and substring algorithms for displaying the percentage of similarity between text documents. The values increasing and decreasing in the parameters of sentences and substring algorithm was clearly observed in the graphical representation of the results. From the results, it is observed that the comparison between the number of sentences and the number of substrings required for identifying the matched items in the documents. The number of sentences increases from 100 to 500 in number, the substring also increases with to the number of documents to be identified for the matching purposes. We can observe a gradual increase in the number of sentences, the increase in the substring can also be observed from time to time.

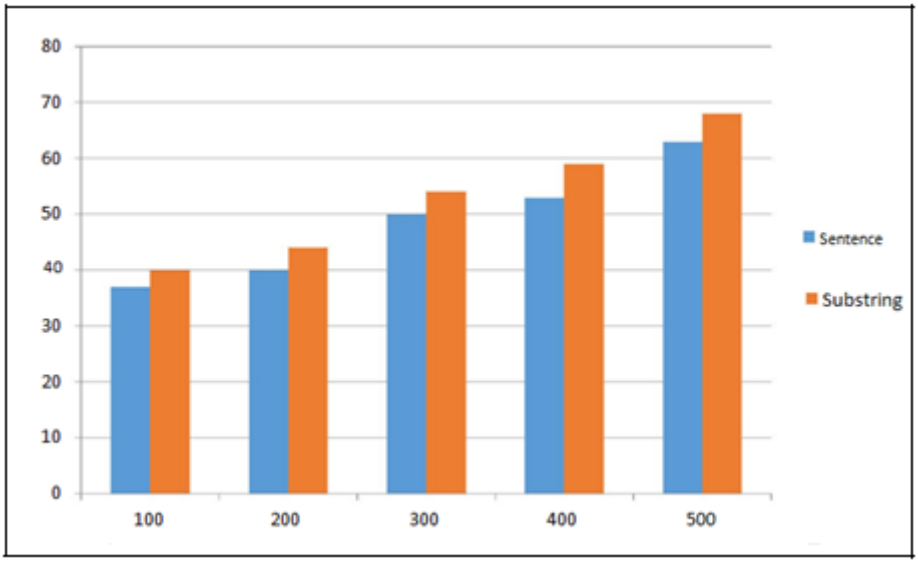


Figure 5. Comparison between Sentence and Substring Algorithms for Displaying % of Similarity

7. Conclusion

The only way to secure the original work and giving proper recognition to it is by copy detection technique. There are many tools available for this copy detection but they all have their own advantages and disadvantages. And the major disadvantage of all is, they

are computationally high cost as the input data set for source documents are increasing extensively. Other drawback is the varying accuracy levels with the change in one simple parameter "tolerance", which is not at all acceptable. Thus, we came with a new approach to the existing problem of copy detection which shoots up to a single character level known as "Substring comparison algorithm", which compares two documents up to character level and gives more accurate output. From the presently developed model gives us the mostly used model for identifying the solution to the existing users for their uses. The present model finds the values from the amount of data that was being copied from the other documents in terms of character level. The present model finds the similarity between the two documents or the document that has to be used to find the amount of data that was being copied from the other documents was verified and displayed at the user end. In the present model, we had identified the similarity level only at the level of characters level. In the coming future, the similarity between the word to word and also the similarity between the images can also be tried to identify. This problem of plagiarism can be reduced to the major of its extent by using these type of model of techniques to identify the similarity.

References

[1] S.Jamal, Plagiarism detection techniques, Master's thesis, COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY, **(2010)**.
[2] M.Z.Eissen, B.Stein and M.Kulig, "Plagiarism detection without reference collections", 30th Annual Conference of the German Classification Society, Berlin, **(2007)**, pp.359–366, Berlin.
[3] N.Shivakumar and H.G.Molina, "Scam: A copy detection mechanism for digital documents", 2nd International Conference in Theory and Practice of Digital Libraries, Texas, **(1995)**, pp. 1–13.
[4] M.Roig, "Avoiding Plagiarism, Self-Plagiarism, and Other Questionable Writing Practices", A Guide to Ethical Writing, Volume 2, St. Johns Univ. Press, **(2006)**.
[5] J.Bloch, "Academic writing and plagiarism: A linguistic analysis", In English for Specific Purposes, Vol.28, **(2009)**, pp.282–285.
[6] I.Anderson, "Avoiding plagiarism in academic writing", In Nursing Standard, vol.23, No.18, **(2009)**, pp.35–37.
[7] K.R.Rao, "Plagiarism, a scourge", In Current Sciences, Vol.94, **(2008)**, pp.581–586.
[8] S.M.Alzahrani, N.Salim, and A.Abraham, "Understanding plagiarism linguistic patterns, textual features, and detection methods", IEEE Transactions on Systems, Man, and Cybernetics, PartC, Vol.42, No.2, **(2012)**, pp.133–149.
[9] G.Salton and M. J. McGill, "Introduction to modern information retrieval", McGraw-Hill,Vol. 2, **(1983)**.
[10] P.Martin, B.Alberto, B.Stein, and P.Rosso, "Cross-language plagiarism detection", In Journal of Language Resources and Evaluation, Vol.45, No.1, (**2011)**, pp.5–62.
[11] E.Stamatatos, "Intrinsic plagiarism detection using character n-gram profiles", 3rd PAN Workshop, Uncovering Plagiarism, Authorship and Social Software Misuse, Donostia, Spain, **(2009)**, pp.38–46.
[12] S.Meyer, Z.Eissen, and B.Stein, "Intrinsic plagiarism detection", European Conference on Information Retrieval, London, UK, **(2006)**, pp. 565–569.
[13] B.Stein, M.Koppel, and E.Stamatatos, "Plagiarism analysis, authorship identification, and near-duplicate detection", In Special Interest Group of Information Retrieval, Vol, 41, No.2, **(2007)**, pp.68–71.
[14] B.Stein, L.Nedim, and P.Peter, "Intrinsic plagiarism analysis", In Journal of Language Resources and Evaluation, Vol.45, No.1, **(2011)**, pp.63–82.

