CBR-based Archiving of Contextual Documents with Check for Plagiarism

Chitrita Chaudhuri [1],
Associate Prof ,Dept. of Computer Science and Engg, Jadavpur University
188 Raja S.C. Mallik Road, Jadavpur, Kolkata-700032, India

Chhanda Roy[2],
IEEE Member & Electrical Engineer from Jadavpur University

Atal Chaudhuri[3]
Professor, Dept. of Computer Science and Engg, Jadavpur University
188 Raja S.C. Mallik Road, Jadavpur, Kolkata-700032, India

Abstract: The present work aims to serve the research community by utilizing Case-based Reasoning (CBR) concepts in Data Mining applications to achieve the following : (1) Mining text documents to collect keywords and storing them in a FP-tree structure to generate frequent key-phrases. (2) Using Open Hashing technique to produce an index for the document from associated keywords, and storing in a variation of Separate Chained data structure to resolve collisions. And (3) checking key-words and key-phrases from temporary indexed store of test documents against key-words and key-phrases from relevant documents to test for plagiarism level. For this purpose, key-phrases, which are assumed to be frequent patterns derived as bi-grams and tri-grams of unique key-words, are mined from the documents using frequent pattern generation algorithms such as FP-tree-growth algorithm and Apriori based algorithms. The superiority of FP-tree-growth is experimentally verified and its affinity with the CBR-strategy well established [4]. The threshold experience level of the CBR-system is preliminarily set by including electronic text-books on relevant topics within the document corpus. To facilitate subject-wise indexing, keywords of the test document are mapped onto an index-base to detect measures which guide the classifier module for both retaining a new case as well as retrieving existing ones from the corpus.

Keywords: Case-Based Reasoning (CBR), Bi-grams, Tri-grams, Frequent Patterns, FP-tree-growth Algorithm, Separate Chaining.

1 Introduction

This paper delves into the rich domain of text mining in determining the subject of a document, so that it can be archived with proper indexing for ready retrieval whenever required. It further explores the possibilities offered by the Case-Based Reasoning (CBR) methodologies for building such an archive, after passing each fresh document through a forensic test to detect plagiaristic attempt, prior to storage. In the researching community, documents need to be archived all the while, but one has to go about the task very cautiously.

Researchers do sometimes adopt unfair means, such as copying. Ideal measure to prevent preservation of duplicated material would be to detect malpractices at the nascent stage through text mining techniques.

Text mining involves some preprocessing on the document repositories in order to effectively search and retrieve information from a reduced volume of a text document [2, 6]. Besides removal of non-ascii components, raw text documents need to be freed of irrelevant words which have little bearing on the subject matter. So such words, commonly known as 'stop-words', need to be removed.

Another common procedure followed in text preprocessing is 'stemming'. In stemming, reduction is affected by preserving only the stem or 'root' word such as 'thank' instead of 'thanking' or 'thanked'. Stemming can be achieved through algorithms such as the one introduced by Porter (1980) [10] or earlier aggressive stemmer developed by Lovins (1968) [5]. A variation of the later known as Iterated Lovins stemmer adopted in [11] may also be used. But the main emphasis of this research being the exploration of the efficacy of the CBR technique in the present domain, such elaborate algorithms have been avoided and a simpler heuristic tried out as explained in sub-section 2.2 [Figure 3] below.

In the same vein, the present researchers avoided using KEA and Keyterm Methods [9] to extract keywords, although prevalently popular in machine learning community. Success of such schemes would be dependant on availability of a large training dataset, and the training time would also be significant. A heuristic technique utilizing TF-IDF score [15] [9] has been tried out with a slight variation imposed on the calculation suggested in [6] which improved the process of extracting keywords as illustrated in sub-section 2.2.

A need for introducing the concept of ***Case-Based Reasoning*** is felt over here. In a nutshell, it is a problem solving paradigm where a problem and its solution forms a pair, identified together as a ***case***, which gets indexed and stored within a memory-***base***. The ***reasoning*** part comes into play, when the case-base gets sufficiently experienced, to provide solutions for new problems. This is achieved by the CBR system through ***retrieving*** similar ***cases*** stored earlier, ***reusing*** solutions of ***retrieved cases*** to generate new solution, in the process ***revising*** new solution if so required, and finally ***retaining*** the new problem solution pair as a new ***case*** or experience, to enrich the knowledge level of the CBR system. This is the CBR-cycle that is so similar to the ***incremental learning*** procedure followed by humans. To the machine learning community, it also poses as a ***lazy learning*** technique of classification, using previously accumulated experience, retrieved as similar cases from its base, as and only when such guidance is needed. The term lazy is in direct opposition to the ***eager learning*** technique of building a classifier model apriori through proper training with class-labeled data, or solved problems, to be more specific [1, 3, 8, 12, 13, 14].

The CBR system here handshakes with the FP-Tree Growth algorithm [7] to generate the bi-grams and tri-grams, which are contiguous sequences of two and three words respectively, that are preserved in order to represent a document in a memory based structure, a variation of the one discussed in [9]. The following section 2 describes these and other methodologies adopted in assessing the test document for literary authenticity before classifying and placing it in the case base corpus. In section 3 is described the experimental setup with specifications of four sets of data used to test the system and the environment under which the testing took place. The next section 4 contains the actual snapshots of the outputs produced by the system while testing it on the four datasets, accompanied by comments reflecting on the inference drawn from these outputs. The final section 5 presents the concluding remarks which also highlight the prospective application area where such a system can pose as a practical

solution.

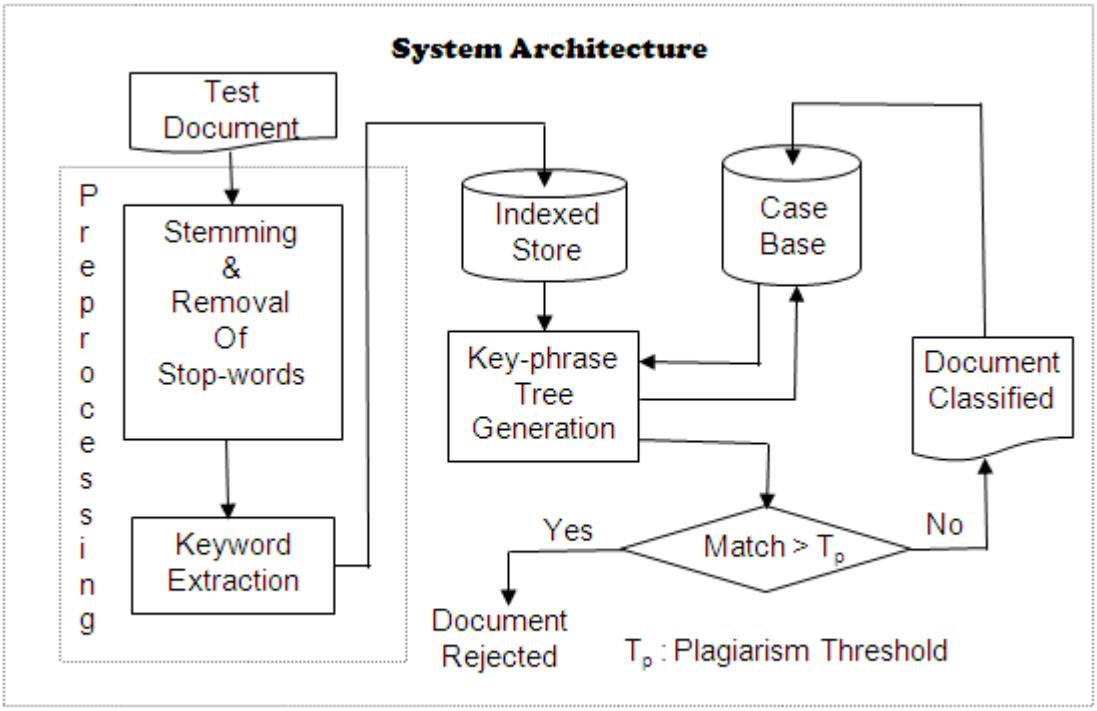


Figure 1 A general schematic diagram of the proposed system.

2 Methodology

The discussion on the methodology begins by presenting a schematic diagram of the proposed system in Figure 1 above. A document on being presented to the system would first be preprocessed. During this preprocessing step, discussed in detail in sub-section 2.2, the keywords are extracted and provisionally accumulated in an indexed storage area. Later, these are utilized to generate key-phrases within a tree structure, which preliminarily serves to detect the domain of the test document. At this stage, this structure effectively represents the test document.

In the next stage the key-phrases are matched with documents preserved in the case-base, to detect whether the test document contains plagiarized excerpts from the domain corpus or not. If the match is more than the plagiarism threshold, then the test document is rejected. Otherwise, the document is found fit to be stored in the corpus as a new case after being appropriately classified, as depicted in Figure 2.

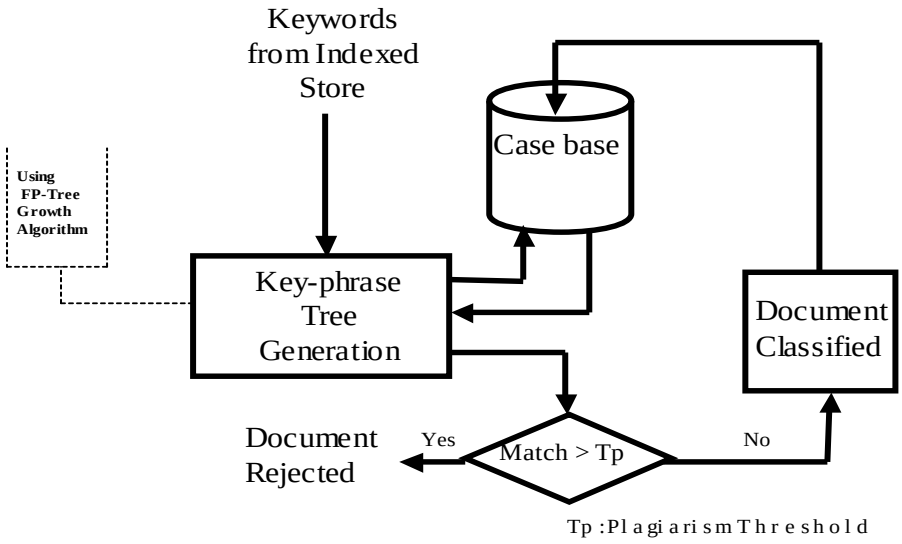


Figure 2 Schematic representation of procedure for extraction of key-phrase.

The details of each component is described in the following few sub-sections. At the beginning is presented the format used to store the words in the indexed base. Next is discussed the process of generating keywords. The third sub-section mentions the different frequent pattern generation techniques, and the results of comparing them temporally, before utilizing the chosen algorithm to extract the key-phrases from the test document. This stage is crucial to attain both the objectives of the present research work : evaluation of the domain of the test document, and successful retention within the case-base representing the domain corpus, after detecting the literary authenticity of the document.

2.1 Data structure techniques for storing words

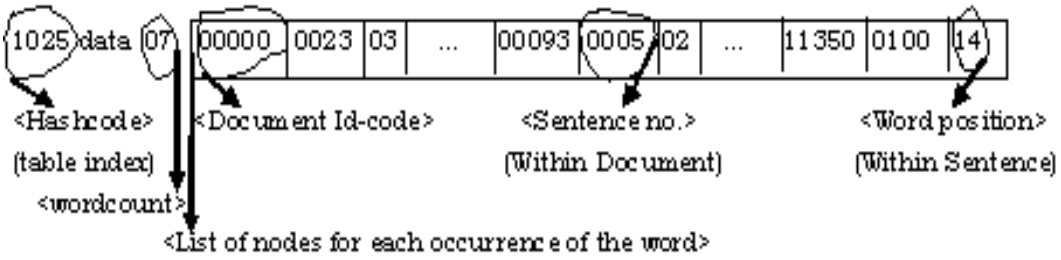
The parsing of sentences and generation of hash code value of each keyword in the text document is discussed in [4].

The hash-code is used as the index of the base table. At each indexed position in the base table is stored the word itself, the total count of that particular word appearing in all the documents in the base and a list consisting of a chain of nodes, each accommodating the following items :

< document-id-code >	*< sentence-no >*	*< word-position >*

The data structure utilized for the word base may be visualized with the help of an instance. The similarity with a vertical format Market-Basket Data is apparent here, with a word in lieu of an item, and the structure of the nodes with the document-id representing the transactions wherein they appear.

E.g. entry for the word ***data*** in the base would be as follows :



Here, the documents referred to in each node are the cases that have already been stored in a properly indexed form within the corpus. The *document-id*s correspond to identification tags that are mandatory for correctly locating each separate document in the corpus. The *sentence-no* is a count maintained to identify that portion of text which can be segregated from the remaining part of the document progressively through punctuations registered with the system. The *word-position* similarly indicates the position of the word within each sentence. These last two elements help in detecting verbatim plagiarism in a new document that arrives for inspection before being added to the corpus.

A temporary word base, of same structure as depicted above, is maintained for such a document under surveillance, till it has overcome the plagiarism check. Once cleared of all charges, the main word base is updated with the content of the temporary base records of all the keywords in the test document. If any of these is a new one, then the temporary base record gets stored permanently as it is. Otherwise the corresponding word records are located and the temporary word counts are added to the existing ones, and the rest accommodated as parts of the existing lists. At the same time, the case base of the corpus gets enriched by allowing the test document as a new entrant under the detected subject category.

The next sub-section describes the procedures adopted for identifying the keywords of a test document.

2.2 Tokenization

Tokenization is a preprocessing step adopted to identify keywords within a document. The tokens or keywords, once identified, serves double purpose in the present context. Firstly they help to determine the topic of the document, and secondly they also assist in tracking literary thefts. The primary goal to be attained at this stage is to detect the root or stem of each word in the text document to reduce redundancy. The process is called stemming, as already mentioned in section 1 above.

Stemming is achieved with the help of a crude heuristic which eliminates common word-ends, maintained in a separate list, when partial matches with the same is encountered within a word. The algorithm for detection of common word-ends and extraction of root-words is presented in Figure 3 below. After applying the procedure, the stemmed words are mapped onto the same root word. The nodes corresponding to the stemmed words, along with their positional records within sentences of the test document, are next necessarily merged in order, reducing time and space complexity for the next stages.

Algorithm: To eliminate common word-end and extract root-word.
Input:

- ***L*, a list of common word-ends;**
- ***D*, the dictionary of collected root-words;**
- ***X*, a string representing the word to be stemmed.**

Output: *Y*, the string representing the stemmed root-word.
Method:

```
(1) Y = X ; // Backup X in Y
(2) for each element l Є L {
(3)     p = position( X, l ) ; // get position of l within X (=0 if not found)
(4)     if (p) // non-zero i.e. l found within X
(5)     {       s = stem(X, p-1 ) ; // get the first p-1 length substring of X
(6)         verify(s); // verification by linguistic expert
(7)         for each root-word r Є D {
(8)             if ( match( s, r ) ) // substring matches current root-word
(9)                 goto step 12 ;
(10)        }
(11)        D = D U s ; // add the new candidate root-word s into the dictionary D
(12)        Y = s;
(13)        goto step 16 ;
(14)    }
(15) }
(16) return Y;
```

Figure 3 Algorithm for extraction of root-word

The sanctity of a root word should be verified by a linguistic expert at step 6 in the above algorithm, by an interactive display of the root word, and a corrected version provided, if necessary, at this stage. For example, the root word extracted from the word 'tried' is 'tri' and should be replaced by the corrected version 'try'.

To further reduce the bulk of the word-base, only the keywords, or the tokens, are sieved out next. The sieving process adopted here to screen the stop words or common words from appearing in the word-base is a very elementary one. The technique utilizes a vector-space model of a document within a corpus. In this model each document can be represented as a vector in k-dimensional space, where k is the set of keywords which is capable of defining the context of a particular set of documents. The strategy is explained in the following part of this sub-section.

Generally there are two types of scores associated with each term in a text document. The first one is the Term Frequency or TF–score which helps in detecting the more frequently used terms in the current document. The second is the Inverse Document Frequency or IDF-score. The combination of the two produces the TF-IDF rating, computed by multiplying term frequency and inverse document frequency, to determine the relevance of a word in a document.

The simplest representation of term-frequency is the frequency of occurrence of the term in a particular document. Hence, when a term does not appear in the document the term frequency with respect to that document would be 0 (zero). Otherwise, if it does appear in the

document, the term frequency may simply be taken to be 1 (one) in a binary mode, or be represented by the cardinality of occurrence. To normalize, the relative term frequency may be calculated as the ratio of the frequency to the total number of occurrences of all the words in a document. The evaluation technique adopted here is based on the Cornell SMART system indicated in [6]. The procedure for calculating TF-score and IDF-score is discussed in a previous paper [4].

In the present scenario, the documents are broadly classified into two categories. Those which deal with topics related to the test documents are the relevant ones, whereas those without any subject bias are called the irrelevant documents.

As the term frequency of a word is computed with respect to all the documents, which include both relevant and irrelevant documents, the terms having high TF in relevant documents and low in irrelevant documents has a high chance of being a key term. And for one having more or less same TF in all documents has a high chance of being a common term. On the basis of this logic, two averages are calculated :

Avg_1 is average TF of a word in relevant documents.

Avg_2 is average TF of a word in irrelevant documents.

A low difference value of these two averages indicates a common term and a high positive difference between Avg_1 and Avg_2 indicates a key term for the relevant topic.

Now as common terms appears in both relevant and irrelevant documents, IDF for those terms will be low. And as the appearance of most key terms in relevant documents have a very low chance of appearing in irrelevant documents, so IDF for key terms will be high.

Hence the TF-IDF score of a term within a document may be calculated as follows :

$$(\text{TF-IDF})_{ij} = (Avg_1 - Avg_2) \times (\text{IDF})_i \qquad (1)$$

A high value would indicate a likely key-term, and a low value indicates a common term.

Equation (1) is used for every term in the document and the frequent terms sieved out by techniques utilized in [4].

2.3 Extraction of Key-phrases

Through experiments carried out during research enclosed in [4], the FP-Tree-growth algorithm has been proved to be the best to extract frequent patterns. These are the frequent bi-grams and tri-grams and thus are the required key-phrases for a test document.

Prior investigation also indicated that FP trees are most effective for storing large stock of keywords and can facilitate the memory based checking scheme adopted here. Each key-phrase, consisting of the contiguous or neighbouring keywords, can be obtained by traversing from the root to a leaf node of the tree. The module is further enhanced by the limiting condition that no path can exceed beyond the length of three since bi-grams and tri-grams are the only ones being considered as key-phrases to be preserved.

In the following Figure 4 is represented the procedure adopted to generate a memory based tree structure for holding the key phrases of a document based on the FP-tree algorithm. The support counts of individual keywords and key-phrases are maintained within the FP-tree structure of a test document in the present scheme. These counts play a vital role to establish some important aspects associated with the document. The more frequent amongst these keywords or key-phrases are utilized to identify the domain of the test

document and thereby generate an appropriate index value for it. This index value can be used to retain the document as a case within the archive-base later. But even before that happens, this index also contrives to locate and retrieve relevant documents within the archive as candidates for matching during plagiarism tests.

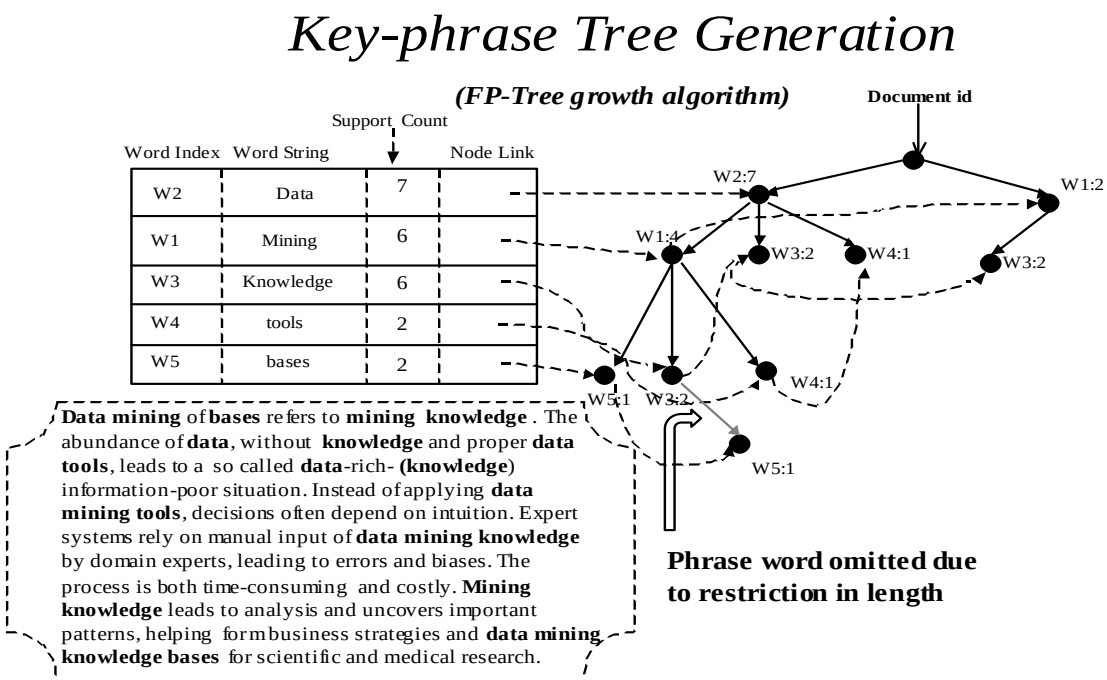


Figure 4 Schematic of Key-phrase Tree Generation using algorithm in Figure 4

While checking for plagiarism, the key-words in the tree table are next used to search the stored case structures. At this stage, the list of triples maintained for each keyword appearing within a key-phrase, both for the test document and the candidate from the archive, are matched in terms of relative positioning within sentences. If the percentage number of times these tally, out of the total number of key-phrases within the test documents, exceeds the plagiarism threshold, then the test documents gets charged and hence rejected. Otherwise, the test document is classified on the basis of its index value and is placed appropriately within the archive.

Intelligent plagiarists often utilize more than one source document. To combat such clever tactics, a separate score is preserved for each case-document, as is depicted in the following Figure 5, in a temporary structure designated as the *Document Index Table*. Another count, maintained at the end of each matched key-phrase path in the test document, is also summed up and checked against the plagiarism threshold. An indirect charge of plagiarism may also be brought if found guilty under such a condition.

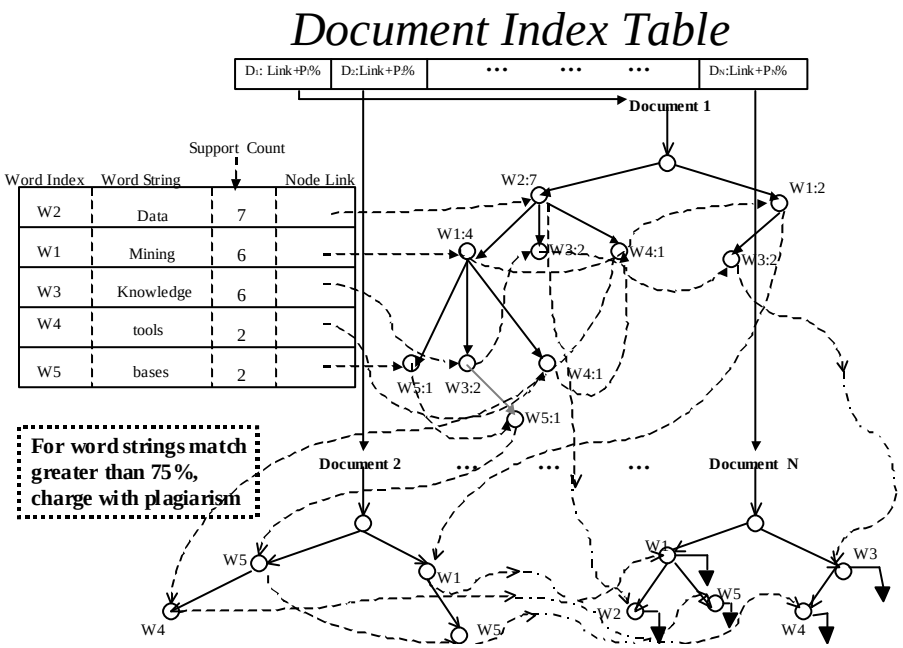


Figure 5 The structure of the Document Index Table with respect to the keyword base.

3 Experimental Setup

The experiments carried out in the earlier context [4] dealt with choosing the correct frequent pattern generation algorithm. The details of the experimental setup for this process and the outcome are provided in [4]. The present phase deals with the performance of the CBR and the setup required for this phase is provided below.

The CBR system developed using the output of the FP-tree-growth algorithm is being checked with some test documents, to prove/disprove its plagiarism detection capabilities. Plagiarism can be detected quite effectively with the help of sample synthetic texts formed by either copying verbatim from a single case document or through cut and paste actions on several such documents. Here four sets of related data-files are used, the details of which is provided below in Table 1. The documents are tested with respect to a case-base corpus which has intentionally been enriched by the main sources of some of the test data-files concocted just for the purpose of evaluating the system.

Table 1 E-document details for evaluating CBR Performance

Data set	***Document Identity***
1	Verbatim copy of introduction part of Chapter 2 of e-book in [1]
2	Inaccurate copy of section 2.1 of e-book in [1]
3	Data set 1 in reversed order
4	Large document on data mining concocted from multiple sources

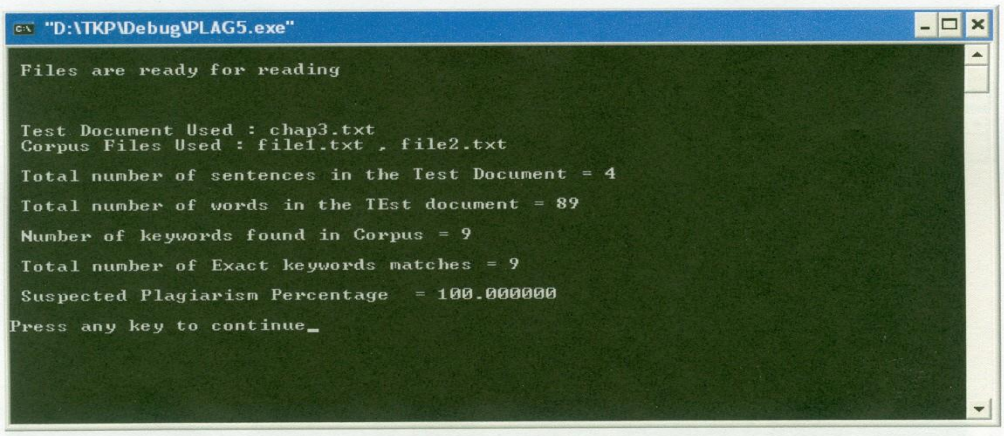
The descriptions of the results and the analysis generated by the system is described under sections 4. All algorithms of the system are developed using C/C++ codes and tested under LINUX/ Windows environment.

4 Results and Performance Analysis

The choice of the FP-tree-growth algorithm is indicated by experiments conducted in [4] and is based on two factors. The first is the performance issue. The system has been found to perform efficiently and is scalable too within the constraints mentioned earlier. The second one involves the storage structure lent by the algorithm to the CBR system utilized in this phase of the present work.

The CBR is designed to manifest three properties in this research work. The first one is its propensity to accommodate the documents of a text corpus – that is its storage capacity. The second is its capability to act as a classifier in order to index and access the right documents in the proper categorical sequence with respect to the domain of the documents concerned. In this phase of the experiment, it is the third property of this particular CBR that gets tested – that of discriminating a test document on the basis of its literary authenticity, prior to allowing its storage within the case-base structure. The first two properties have been touched upon in the context of the Document Index Table and its structure depicted in Figure 5 in sub-section 2.3.

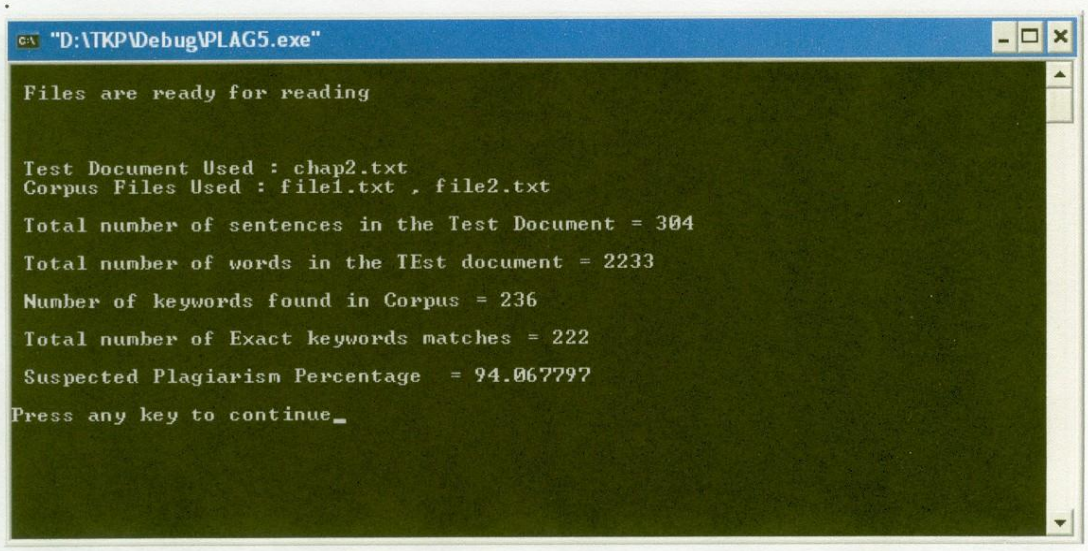
Test Data Set 1: Verbatim copy of introduction part of chapter 2 of the e-book of *"Data Mining Concepts and Techniques"*-Han and Kamber



Snapshot 1: Testing a document which is a verbatim copy of part of the corpus

The above snapshot of the run result with the first dataset in this current phase of experiments establishes the capability of the CBR system, ostensibly named PLAG5, by detecting verbatim copy of a portion of the corpus. Be it made clear that the whole of Chapter 2 of the referred book [6] had already been kept within the case-base corpus to facilitate ready detection of plagiaristic activity. Runtime statistics shown in Snapshot 1 conveys that the first test document is a small one consisting of only 4 sentences having 89 words in all, out of which the process of tokenization generated only 9 keywords. Being a verbatim copy, the total number of matched keywords also tallies to 9, resulting in a straightforward detection of 100% plagiarism. However the result proves the power of detecting verbatim copies.

Test Data Set 2: Inaccurate copy of section 2.1 of the e-book of *"Data Mining Concepts and Techniques"*-Han and Kamber



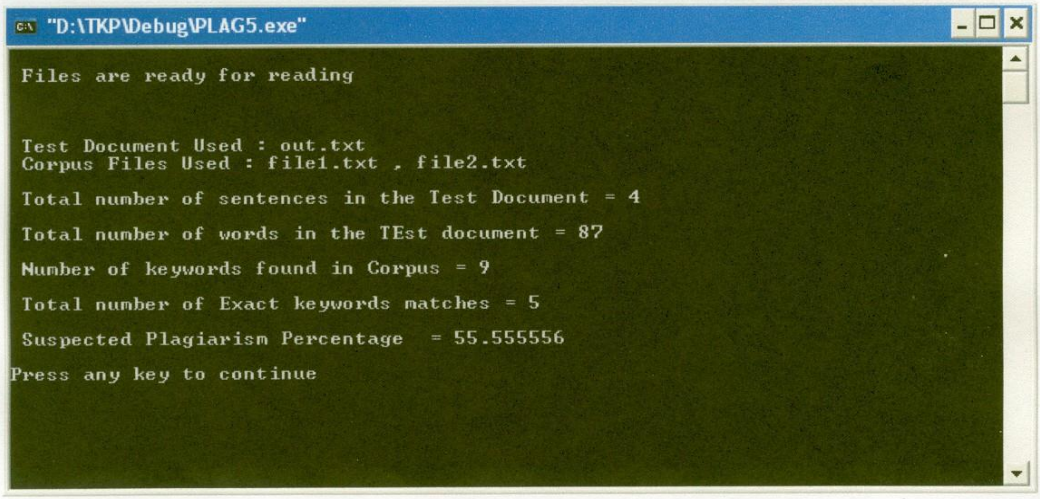
Snapshot 2: Testing a document which is an inaccurate but near enough copy of part of the corpus

The snapshot 2 above indicates high capability of detecting plagiarism even for large datasets such as the second one.

In fact, the following snapshot 3 also corroborate PLAG5's capability to detect most types of partial copies.

Since the plagiarism threshold is set to 75% for the present system, the third run-result exposes an interesting characteristic even for the small dataset being tested – the keyword match is approximately halved giving rise to a 50% plagiarism charge. So actually this particular document passes the plagiarism charge, although all the keywords are there. Conclusion is that the significance of the ordering of the keywords is being taken care of by the system.

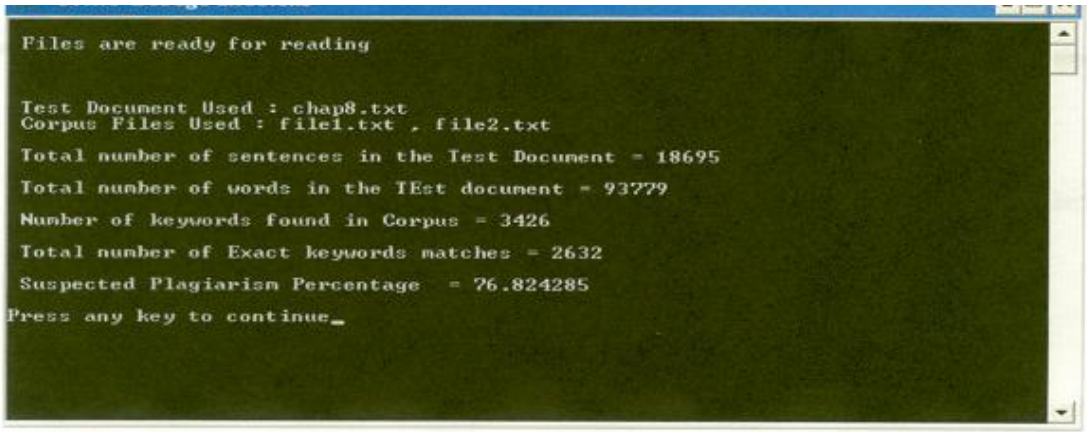
Test Data Set 3: Test data set 1with the words in reversed order



Snapshot 3: Test document in reverse word order of test document 1

The last dataset depicted in Snapshot 4 is a huge one, containing 18695 sentences, 93799 words, and 3426 keywords, out of which 2632 got matched with the corpus. This involves partial copies from multiple documents in the domain. The result of 76.8% plagiarism charge corroborates PLAG5's success in establishing all types of plagiarism possibilities.

Test Data Set 4 : A large concocted dataset containing plagiarised data from multiple documents included within the Data Mining Corpus used for the experimental setup.



Snapshot 4: Testing a document on Text Mining with the Corpus of Data Mining

5 Concluding Remarks

The time complexity of the FP-tree growth algorithm has been found from earlier experiments [4] to be scalable irrespective of test-document size and number of keywords and key-phrases extracted. Further benefits accrued from the application of this algorithm, by the CBR system, is its efficient data structure and preserved support count values which help to generate a context-sensitive index for the document, preliminarily. The structured output of the algorithm helps to enrich the CBR corpus and the base of keywords to a large extent. This also empowers the system additionally while classifying a new document and establishing its literary authenticity.

On rating the CBR system, the different aspects of the system can be inspected severally. The problem part of the case here is posed as an unchecked and un-indexed document, and the solution is the plagiarism-test imposed on it, as well as the generation of a suitable index value. The retrieval of the set of similar documents from the case-base is facilitated by proper indexing provided at the outset. The reuse and revision of the retrieved documents' case-history, in the form of matching of saved key-phrase patterns, leads to either retaining the document under trial as a new case, or rejecting it on plagiarism charges.

The system also supports the incremental learning procedure which is the hallmark of CBR. In fact, it is this feature of the system which lends itself so suitably to the principle of archiving. Any document presented is structured, indexed and checked automatically under the current scenario – and in the process the case-base also upgrades itself with the new case information. The environment perfectly blends with any research community, where such vigilance on the constant in-flow of material may make life a lot easier for all concerned. The presence of a safe-guard may also contribute towards inculcating stricter discipline and honesty levels amongst the fresh entrants in the arena.

References

[1] A. Aamodt and E. Plaza, 1994, "CBR: foundational issues, methodological variations and system approaches", AI Communications, vol. 7, no. 1, pp. 39-59

[2] Antonina Kloptchenko, 2003, "Text Mining Based on the Prototype Matching Method", Turku Centre for Computer Science, TUCS Dissertation, No 47.

[3] C. K. Riesbeck, R. C. Schank, 1989, "Inside case-based reasoning", © Lawrence Erlbaum Associates, Inc.

[4] Chitrita Chaudhuri and Atal Chaudhuri, 2011, "Detection of Verbatim or partial Duplication from Multiple Source Documents using Data Mining Techniques and Case-Based Reasoning Methodologies", ", EAIT 2011, Kolkata, pp. 129-132.

[5] J.B. Lovins, 1968, "Development of a stemming algorithm", Mechanical Translation and Computational Linguistics 11, 22-31.

[6] J. Han, and M. Kamber, 2001, "Data Mining Concepts and Tecniques," Morgan Kaufmann Publishers, SanFrancisco, USA, ISBN 1558604898.

[7] J. Han, et al, 2000, "Mining frequent patterns without candidate generation",In Proc. 2000 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'00), pp 1-12, Dallas, TX

[8] J. Kolodner, 1993, "Case-Based Reasoning", Morgan Kaufmann

[9] LI Juanzi, et al, 2007, "Keyword Extraction Based on tf/idf for Chinese News

Document", Wuhan University Journal of Natural Sciences, Vol. 12, No. 5, doi: 10.1007/s11859-007-0038-4.

[10] M.F. Porter, 1980, "An algorithm for suffix stripping", Program, 14 no. 3, pp 130-137.

[11] P. Turney, 1997, "Extraction of Keyphrases from Text : Evaluation of Four Algorithms", Technical Report ERB-1051 (NRC-41550), Institute for Information Technology, National Research Council of Canada, Ottowa, ON, Canada.

[12] R. C. Schank, 1982, "Dynamic Memory : A theory of reminding and learning in computers and people", Cambridge, UK: Cambridge University Press

[13] R. C. Schank, 1984, "Memory-based expert systems", Technical Report (# AFOSR. TR. 84-0814), Yale University , New Haven, USA

[14] Watson and F.Marir, 1994, "Case-Based Reasoning : A Review",The Knowledge Engineering Review, Vol 9, No. 4.

[15] Y. Zhang, et al, 2005, "Narrative Text Classification for Automatic Key Phrase Extraction in Web Document Corpora",WIDM'05, Bermen Germany.