A Document Comparison Approach using Hybrid Keyword and Structured Full Text Vocabulary Searches

Kudachamai Boonsuk
Technopreneurship and Innovation Management
Program, Graduate School
Chulalongkorn University, Bangkok 10330, Thailand
Looktaomis2011@hotmail.co.th

Peraphon Sophatsathit
Advanced Virtual and Intelligent Computing (AVIC)
Center, Department of Mathematics, Faculty of Science
Chulalongkorn University, Bangkok 10330, Thailand
Peraphon.S@chula.ac.th

Abstract— **This paper proposes a systematic full text search on document using a combined keyword and structural similarity of documents under consideration. The approach operates in two steps. The first step uses a set of designated keywords to acquire potential desired documents by means of an open source tool. The second step builds a suffix tree of frequently used vocabulary to retrieve the most similar documents from the acquired documents. In so doing, variations on contextual matching of full text search can be mitigated, wherein the resulting performance turns out to be quite acceptable. The ultimate goal is to arrive at a platform independent full text search technique that can be realized. The benefits for this scheme are two folds. On the one hand, relevant document can be retrieved as close to the desired document as possible. On the other hand, suspect plagiarism can be identified to some extent, which is dependent on the effectiveness of the proposed approach with plenty of rooms for future improvement. The proposed work will eventually be put to real use for database retrieval in a small business enterprise.**

Keywords- full text search; structural similarity; suffix tre;, contextual matching; plagiarism.

I. INTRODUCTION

All text-based documents written in prose are naturally composed of sentences and paragraphs. Archiving these documents is usually accompanied by keywords, whereby subsequent reference or use can quickly retrieve the desired document from the archive. This keyword mechanism is well entrenched in information retrieval process from the dawn of information storage and retrieval technology. The rationale is simple: users can just provide the right keywords in order to precisely retrieve the desired document. Despite tremendous assistance by many powerful search engines, this keyword search poses some formidable and recalcitrant improvement challenges. The advent of various full text search schemes introduces a promising simplified document lookup process. Nevertheless, the techniques still do not differentiate much from its keyword-based forerunner in terms of retrieval precision and speed. The sheer volume of text to be matched so as to extract the closest, or exact in ideal case, desired document makes it unlikely to carry out successfully.

There are four major factors that must be resolved in full text search. First and foremost, the original document is usually created and stored in unstructured format. There is no discernable way, aka record format, to systematically locate it in the voluminous archive. Secondly, lengthy sentences of unanticipated counts (though in practice are limited) are in most cases not suitable for composing necessary SQL conditions. Thirdly, most database management systems are built to support keyword match. The newly concocted full text search modules (as an enhancement for latest release) are not compatible with older releases and legacy applications, not to mention the high cost of upgrade. Finally, the forefront DBMS full text implementations are proprietary, employing different approach and platform specific. Users and developers alike must tailor their applications to comply with the individual platform, hence less portability and more customization.

Bearing the aforementioned problems and issues, the proposed research introduces a concise novel algorithmic approach to be implemented on an open source tool and well known suffix tree structure that allow users and developers to utilize the full text search module in a straightforward fashion. The proposed approach will be elucidated as follows. Section 2 recounts some directly impact related works through which the proposed approach is derived. Section 3 describes the novel two-step proposed approach. Section 4 demonstrates the viability of the proposed approach via a series of experiments. Some benefits and shortfalls are discussed in Section 5. Section 6 expresses a few final thoughts and future enhancement.

II. RELATED WORK

Varelas [1] used hypernym and hyponym to investigate word search based on semantic similarity, focusing only on noun and verb as the search basis. Xie [6] explained the quality dimensions of Internet search engines. Goldman [11] employed proximity search to retrieve information and ranked the outcome by score. This technique is exploited in this research to compute word proximity, wherein search efficiency can be objectively measured. Salton [2] proposed term weighting technique to retrieve the desired text, while Hofmann [12] employed PLSA technique to compute term weight. McCandless [5] explained the use of Lucene, an

978-1-61284-840-2/11/$26.00 ©2011 IEEE

open source tool being applied to keyword search in this work. Chartbunchachai [8] suggested weighted positional measure of characters and words in string similarity matching scheme which was also adopted in this work. The main referential work has been taken from Pukkasenungi [10] where text matching is categorized and evaluated. The five categories being applied are illustrated in Figure 1. The same discriminating criteria are strictly followed, i.e., exact match if cosine of similarity is equal to 1.0, plugin ≥ 0.8, subsume ≥ 0.5, partial ≥ 0.3, and fail ≤ 0.1. Should the cosine fall between $0.1 < \cos\theta < 0.3$, matching process would be re-evaluated accordingly.

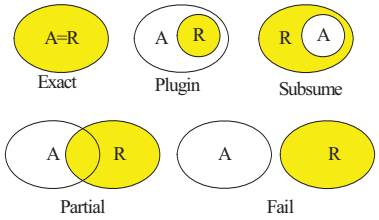

Figure 1. matching categories.

Performance measure can be carried out in many ways. Makhoul [4] suggested their approach for information retrieval. Balinski and Danilowicz [14] suggested re-ranking of inter-document distance method according to the ideal document. We resort to simple scoring (weight-distance [7]) technique and measure the precision/recall/F-measure to gauge the performance of the first step selection or hereafter referred to as *rough pick*. Meanwhile, word proximity forms a suffix tree [9] at which the root word is determined according to the matching category or *fine pick*.

III. PROPOSED APPROACH

The proposed approach is divided into two steps, namely, a keyword search front-end and an algorithmic procedure to gather relevant document and handle full text search, respectively. The front-end step is carried out by means of an open source tool called Apache Lucene, hereafter referred to as Lucene for brevity. The second step builds a suffix tree to analyze full text similarity based on pre-established criteria.

A. Front-end keyword selection process

The procedure begins in the same manner as traditional word search. A set of single term words are collected from WordNet [3] as a preliminary matching process. The words, their referential document, and associated dimension or weight form a vector space model for the search process. Word lookup is carried out by comparing term frequency–inverse document frequency (tf-idf) weighting value of the target vector holding the desired word with the source vector being retrieved from the vector space. A matching indicator based on similarity cosine is computed to determine if a document is the closest one using document relevant (dr) and document frequency (df) having high discriminating power.

The documents obtained by keyword search may not represent all the desired information. Typically, they can be classified into four groups as follows:

1. retrieved and relevant
2. not retrieved but relevant
3. retrieved but irrelevant
4. not retrieved and irrelevant

Based on the above classification, it is impractical to retrieve any 100% compatible documents as desire. We will thus employ three statistical classification indicators for the proposed approach performance measurement, namely, precision, recall, and F-measure which are defined below.

$$Precision = \frac{|\{relevant\ doc\} \cap \{retrieved\ doc\}|}{|\{retrieved\ doc\}|}$$

$$Recall = \frac{|\{relevant\ documents\} \cap \{retrieved\ documents\}|}{|\{relevant\ documents\}|}$$

$$F\text{-}measure = \frac{2 * precision * recall}{(precision + recall)}$$

B. Similarity analysis process

Matching of full text or long character string has been a challenging recalcitrant text search research. Two obvious non-candidate of matching categories are exact and fail matches as they are impractical to undertake. Our compelling question is how to adapt some of the prior works to determine if either plugin, subsume, or partial matching of the search string is an acceptable outcome. A preliminary investigation was conducted based on weighted positional character matching proposed by Chartbunchachai [8] to establish a tentative minimal threshold length for further in-depth investigation. Our approach thus begins with the original document by tallying a list of most frequently used words and create a corresponding suffix tree rooted on the highest frequency word. All succeeding words are sorted, whereby the first *N* words (decided by the users themselves) will form a list of candidate words for subsequent suffix tree construction. The distance of each word from the root word delineates the length of the suffix tree branch. The position of each root word in the original document denotes the height level of the tree. This process is repeated for every target document so that the output suffix trees can be compared for their similarity. The by-product of this work is plagiarism detection [13] which is subject to further improvement. Details on implementation will be furnished in the next section.

C. Document comparison algorithms

The proposed approach was designed to operate in a somewhat different environment setting from conventional DBMS and existing search engines. In most DBMS, searching must be prepared and carried out by the user with the help of standard SQL commands. Search engines, on the contrary, are ready-to-use application software operating through various client browser platforms. The distinction between these two entities is the level of application abstraction. The former relies on the computation power of limited standard SQL vocabulary, while the latter is a

proprietary product by which users must follow available options and operating procedures. We recognize these shortcomings and resort to an open source tool that lies one layer above DBMS, yet still below those handy search engines. This is referred to as the application layer. The advantages so envisioned are three folds. First, users can operate independently of the supporting DBMS. Second, customization of user's application can be done by the user themselves. And last, it's an open source tool that frees the user from any legal infringements.

Bearing the above prospect in mind, the full text search framework is established in the application layer as follows:

1. set up an index creation preliminary based on user's search requirements, i.e., looking for the desired document according to some predetermined keywords, or looking for similar target documents to the original document.
2. create a list of working indices using Lucene library and API [1] to transform source data into Lucene index and document frequency (df) to facilitate subsequent keyword search. All word prefixes and suffixes are handled by Lucene facilities.
3. search and collect the most relevant documents according to the above indices. This collection of documents will be referred to as a document pool. This is the first step or rough pick.
4. in case of finding similar target documents, build a suffix tree based on the original document. The algorithm proceeds as follows:

4.1 create and sort in descending order a list of most frequently occurring words in the original document. The maximum number of words in this list is set to 25 in this study.

4.2 build a suffix tree starting with the first occurrence of the most frequently occurring word as root.

4.3 scan all words that appear to the left of the first occurrence of the root word. If the word is in the sorted list, attach it to the tree and record its position in the document.

4.4 proceed to the next occurrence of the root word and repeat the above scan-attach word to the suffix tree.

4.5 should there be any words in the list left after the last occurrence of the root word, that is, all words in the sorted list that appear to the right of the last root word, attach them to the tree and record their corresponding position.

4.6 for each target document, compute the statistical classification indicators to select the candidate target document out of the document pool. Repeat until all documents in the pool are examined.

4.7 for each document in the candidate pool, build a corresponding suffix tree in the same manner as the original document.

4.8 compare the similarity of the suffix tree obtained from the original document with that of each target document using vector comparison. Apply the above five matching categories to determine the degree of similarity.

5. in case of finding the desired document using predetermined keywords. The algorithm proceeds as follows:

5.1 build a suffix tree based on term weight assigned by the user. The highest term weight will take the root position in the same fashion as the most frequently occurring word from the original document.

5.2 proceed in the same manner as those of step 4.6-4.8.

IV. APPLICATION EXPERIMENTS

To validate the viability of the proposed approach, an actual implementation was conducted via a full-fledged development process. Our challenging mission is to stay focus on full text domain of application to suit the contextual requirements. By virtue of Lucene library support, design activities were minimal once system analysis was completed. The application was written in Java and installed for a pilot run. In the meantime, cross-application APIs were developed to partly fulfill the application layer functionality.

One issue that precipitates from such a framework is design of software API for tool integration using Lucene libraries. The experimental runs were executed following the processing sequence as depicted in Figure 2. Three sets of different document sizes were employed to test the proposed algorithms. They are:

1. small sized document having less than 100 words
2. medium sized document having between 500-1000 words
3. large sized document having more than 1000 words

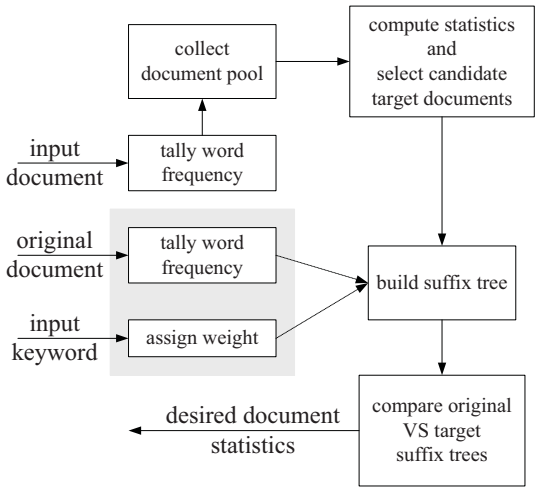

Figure 2. experiment processing sequence

The original document underwent Lucene classification process to create the document frequency (df) for subsequent keyword search. Table 1 shows a sample document frequency list obtained from the original document.

TABLE I. SMALL SIZED DOCUMENT FREQUENCY

Terms	Frequency of occurrence
basic	1
linguistic	1
assumption	1
proximity	3
searching	1
words	7
document	2
implies	1
relationship	2
between	2
given	1
authors	1
documents	1
try	1
formulate	1
sentences	2
which	1
contain	1
single	1
idea	1
cluster	1
related	2
ideas	1
within	3
neighboring	1
organized	1
paragraphs	1

From the above list, the word "words" was selected as the root word for subsequent suffix tree construction. Its closest relatives are "*proximity*" and "*within*" which were used as the word list of the original document. Other words were not used as they would add unnecessary dispersion to the span of the resulting suffix tree. Table 2 shows the statistics of word list position distribution from the root of the suffix tree.

The first set, or small size document, yielded the following statistics shown in Table 3, 4, and 5 which depict word list position distribution, statistical classification indicators, and similarity comparison, respectively.

TABLE II. WORD LIST AND POSITION DISTRIBUTION OF ORIGINAL DOCUMENT

Root \ Term	proximity	within
1	3, 1	-
2	8, 6	-
3	34, 32	12, 3
4	42, 40	20, 11
5	49, 47	27, 18
6	59, 57	37, 28
7	73, 71, 10	51, 42, 13

TABLE III. WORD LIST AND POSITION DISTRIBUTION OF TARGET DOCUMENT

Root \ Term	proximity	within
1	3, 1	-
2	8, 6	-
3	34, 32	12, 3
4	42, 40	20, 11
5	49, 47	27, 18
6	59, 57	37, 28
7	71, 69, 8	49, 40, 11

TABLE IV. STATISTICAL CLASSIFICATION MEASURES OF TARGET DOCUMENT

doc	1	2	3	4	5	6	7	8	9	10
Qt	F	F	F	F	F	F	F	F	F	F
df	8	10	5	4	1	20	3	7	4	20
Pre	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Rec	.75	1.0	1.0	1.0	1.0	1.0	1.0	.71	1.0	.60
F-m	.85	1.0	1.0	1.0	1.0	1.0	1.0	.83	1.0	.75

Qt=query tool, F=full text

TABLE V. SIMILARITY COMPARISON OF TARGET DOCUMENT

	Original doc weight vector	Target doc weight vector	similarity
proximity	15	14	93.33%
within	11	9	88.81%
average	13	11.5	91.07%

We further concocted some candidate documents by arbitrarily substituting or deleting approximately 20 words to mimic plagiarism attempts. The corresponding results to those of Table 3, 4, and 5 were shown in Table 6, 7, and 8, respectively. A comparative plot is depicted in Figure 3.

TABLE VI. WORD LIST AND POSITION DISTRIBUTION OF MODIFIED DOCUMENT

Root \ Term	proximity	within
1	2, 1	-
2	6, 5	-
3	32, 31	12, 3
4	40, 39	20, 11
5	47, 46	27, 18
6	56, 55	36, 27
7	69, 68, 9	49, 40, 12

TABLE VII. STATISTICAL CLASSIFICATION MEASURES OF MODIFIED DOCUMENT

doc	1	2	3	4	5	6	7	8	9	10
Qt	T	T	T	T	T	T	T	T	T	T
df	50	30	40	20	50	70	60	40	20	25
Pre	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Rec	.90	1.0	1.0	1.0	1.0	1.0	1.0	.92	1.0	.68
F-m	.94	1.0	1.0	1.0	1.0	1.0	1.0	.96	1.0	.80

Qt=query tool, T=term

TABLE VIII. SIMILARITY COMPARISON OF MODIFIED DOCUMENT

	Original doc weight vector	Target doc weight vector	similarity
proximity	15	5	33.33%
within	11	8	72.72%
average	13	6.5	53.03%

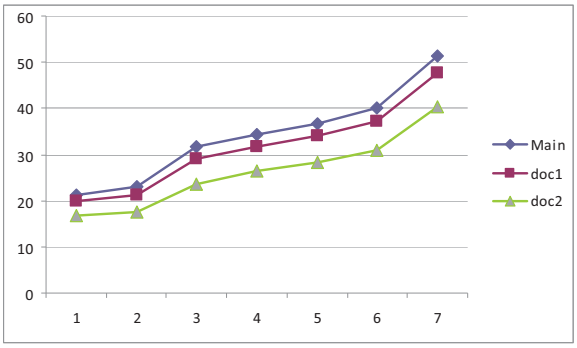

Figure 3. comparative plot of Main(original), doc1(target), and doc2(modified) small sized documents

It is apparent that the original and target documents are closely resemble, whilst the modified document still exhibits similar characteristic to the original counterpart. This reaffirms the similarity measure precipitated from the proposed approach. Similar results were also obtained from medium and large size documents, where 50 and 100 words were altered as depicted in Figure 4 and 5, respectively.

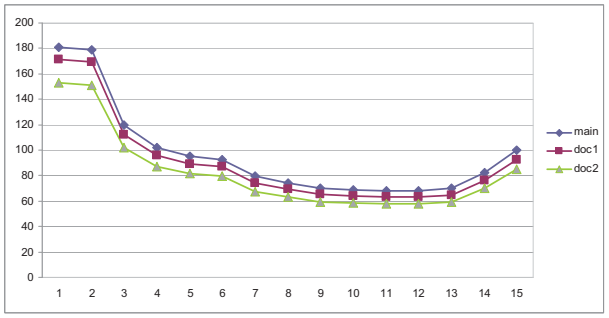

Figure 4. comparative plot of Main(original), doc1(target), and doc2(modified) medium sized documents

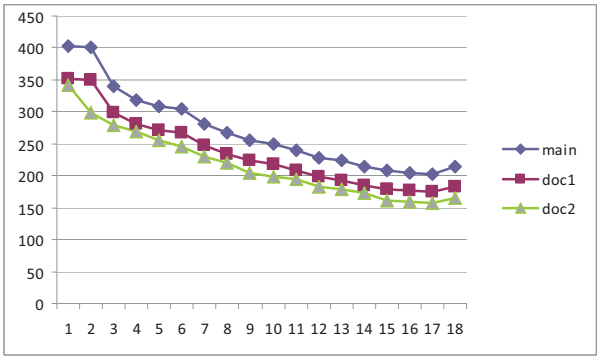

Figure 5. comparative plot of Main(original), doc1(target), and doc2(modified) large sized documents

V. DISCUSSION

From the experimental results, the proposed approach yields the similarity characteristic that renders detection to be straightforwardly determined. Despite subsume matching is prevalent as shown in Table 9, the ease of application makes the proposed approach suitable for typical casual usage in many regards. However, the fact that the shape of the suffix tree depends largely on the frequency of occurrences of the root word renders a wider span but dense suffix tree. In particular, if most words in the list have high *df* value, the similarity measure will decrease considerably.

One major caveat of the proposed approach is root word selection. In the small sized document case, a few root words in the concocted document were haphazardly (due to randomness) replaced by newly introduced words. Not only the frequency of occurrences was reduced, but also the position distribution was affected. The resulting suffix tree and similarity comparison were thus degraded as depicted in modified small document percentage. On the brighter side of this predicament, if the new words fall out of the root word positions, the shape of the suffix tree remains relatively unaffected, and hence yields the same similarity comparison.

Further comparison with commercial software was not performed due to copyrights and the cost incurred from the proprietor.

TABLE IX. DOCUMENT SIMILARITY COMPARISON AVERAGE

Small		Medium		large	
orig	mod	orig	mod	orig	mod
52.38	28.58	61.56	58.57	60.81	55.02

VI. CONCLUSION AND FUTURE WORK

The myriad of documents so generated in today's world, which are further spreaded by the Internet, have dwarfed the effort to locate and retrieve the desired document in an efficient manner. As they are available in more and more format varieties, the predominantly preferred and inexpensive format remains to be text document. In this work, we merely demonstrate a novel two-step full text search approach using structural configuration of the keywords as the principal basis. This serves as a rough pick to ease the recalcitrant full text search problem. The proposed full text search can not only retrieve relevant document effectively by means of a systematic similarity comparison method, but also turns the application around to operate as a plagiarism detection vehicle. We envision that this straightforward implementation will instill further improvement on more efficient and elaborated algorithmic procedures, whereby the notion of full text search can be extended to other forms of document for better and accurate information retrieval.

REFERENCES

[1] G. Varelas, E. Voutsakis, P. Raftopoulou, E. Petrakis, and E. Milios, "Semantic similarity methods in wordNet and their application to information retrieval on the web", Proceedings of the 7th Annual ACM International Workshop on Web Information and Data Management (WIDM'05), Bermen, Germany, November 5, 2005, pp. 8-54.

[2] Gerard Salton, C.B., "Term-weighting approaches in automatic text retrieval", Information Processing & Management, 1988, vol. 24, issue 5, pp. 513-523.

[3] http://wordnet.princeton.edu/, last accessed on November 23, 2010.

[4] John Makhoul, Francis Kubala, Richard Schwartz, and Ralph Weischedel, "Performance measures for information extraction", Proceedings of DARPA Broadcast News Workshop, Herndon, VA, February 1999, pp. 1-4.

[5] Michael McCandless E.H. and Otis Gospodnetić, Lucene in Action, Second Edition, Manning Publications Co, 2004.

[6] M. Xie, H. Wang, and T.N. Goh , "Quality dimensions of Internet search engines", Journal of Information Science 24 (5), 1980, pp. 365–372.

[7] Nalinee Sophatsathit and Peraphon Sophatsathit, "Filtering Search Document using Weight-Distance Transformation", Proceeding of the 13th National Computer Science and Engineering Conference (NCSEC2009), Montien Riverside Hotel, Bangkok, November 5-6, 2009, pp. 389-393.

[8] Nawaphorn Chartbunchachai, Autcha Mutchalintungkul, and Peraphon Sophatsathit, "ANSL Algorithm for String Similarity Matching", Proceedings of the 2nd International Conference on Knowledge and Smart Technologies (KST2010), Burapha University, Chonburi, Thailand, July 24-25, 2010, pp. 54-57.

[9] Oren Zamir and Oren Etzioni, "Web Document Clustering: A Feasibility Demonstration", Proceedings of the 21st annual international ACM SIGIR Conference on Research and Development in Iinformation Retrieval, Melbourne, Australia, August 1998, pp. 46-54.

[10] Pensri Pukkasenung, Peraphon Sophatsathit, and Chidchanok Lursinsap, "An Efficient Semantic Web Service Discovery Using Hybrid Matching", Proceedings of the 2nd International Conference on Knowledge and Smart Technologies (KST2010), Burapha University, Chonburi, Thailand, July 24-25, 2010, pp. 49-53.

[11] Roy Goldman, N.S., Suresh Venkatasubramanian, and Hector Garcia-Molina, Proximity Search in Database, 1998.

[12] T. Hofmann, "Probabilistic Latent Semantic Analysis", Proceedings of the 22nd Annual ACM Conference on Research and Development in Information Retrieval, Berkeley, California, ACM Press, August 1999, pp. 50-57.

[13] Zhan Su, Byung-Ryul Ahn, Ki-Yol Eom, Min-Koo Kang, Jin-Pyung Kim, and Moon-Kyun Kim, "Plagiarism Detection Using the Levenshtein Distance and Smith-Waterman Algorithm", Proceedings of the 3rd International Conference on Innovative Computing Information and Control (ICICIC '08), 2008, pp. 569-569.

[14] Jaroslaw Balinski and Czeslaw Danilowicz, "Re-rangking method based on inter-document distances", Information Processing and Management, Volume 41, Issue 4, July 2005, pp. 759-775.