Linearithmic Corpus to Corpus Comparison by Sentence Hashing Algorithm SHAPD2

Dariusz Ceglarek
Poznan School of Banking
Poznan, Poland
Email: dariusz.ceglarek@wsb.poznan.pl

***Abstract*—This work presents an innovative method of comparing sets of textual documents with an aim to identify common phrase sequences. The $SHAPD2$ (Sentence Hashing Algorithm for Plagiarism Detection 2) algorithm was designed to achieve the goal of a single-pass corpus to corpus comparison. The algorithm was developed taking into account results and observations from previous research activities. It is a highly efficient solution that finds application with considerable amounts of data and excels over other approaches. One of its possible applications is detection of potential plagiarisms comparing not a document against a corpus, but corpus to corpus. Algorithm's performance allows for applications in situations where results have to be served an instant after issuing a query. This makes the $SHAPD2$ algorithm a valuable alternative to the available solutions.**

***Keywords*—document comparison, plagiary detection, longest common subsequence, sentence hashing, Natural Language Processing, text mining, pattern matching**

I. Introduction

This article presents results of the research aimed at designing a novel algorithm capable of robust detection of common subsequences, which is more efficient than existing methods based on sentence hashing [12], [28]. The information retrieval task in which it is more efficient solution is to compare 2 large corpora of text documents in a sequential matter – a corpus of suspicious documents against a corpus of source documents (originals) – in order to detect possible plagiarism attempts. The $SHAPD2$ builds upon earlier research activities which produced good results. These solutions were part of research on Semantically Enhanced Intellectual Property Protection System (SeiPro2S)[6]. The final architecture of SeiPro2S and its functionality has been described in [11]. The novel algorithm proves to be effective in the task defined above, achieving the new level of applicability in previously unreachable scenarios. Such performance was verified throughout experiments considering millions of document-to-document comparisons.

The $SHAPD2$ algorithm operates on hash-sums that represent individual sentences. A similar method was used in a number of other known algorithms working with n-gramms, such as w–shingling [22], minhash [3], simhash [12], etc. It also includes findings from author' previous developments, namely the $SHAPD$ algorithm[8]. However, thanks to improvements, code optimization and introducing new approach, $SHAPD2$ is a new solution, described in detail in the following sections.

II. Related work

The $SHAPD2$ algorithm allows for a robust and a resilient computation of the longest common subsequence shared by one or many input documents. The $SHAPD2$ algorithm processes documents by dividing them into a stream of sentences, where unnaturally long sentences (enumerations, itemizations, etc.) are handled by a special procedure [8]. Such an approach allows to extract extremely long sentences from paragraphs and process them individually.

The process is driven by a modular additive hashing. Hashing is a commonly used technique in Natural Language Processing (NLP) and Information Retrieval (IR) tasks that is used in order to achieve faster word retrieval. In plagiarism detection it is crucial, however, to identify and match longer common word sequences (with special focus on sentences) function with collision lists. Every concept (term) in a sentence is hashed by assigning a number from a previously defined range (during the experiments the limit was set to a large prime number). Furthermore, the individual hashes are summed to represent a sentence. Thanks to the additive nature of the hashing function, sentences with a changed concept order are treated as equivalents. A collision of sentence hashes, where the individual concepts are assigned natural numbers is negligible. Thus, the resulting algorithm not only finds the longest common sequences, but also the longest common quasi-sequences (allowing minor editing changes such as syntactic changes, insertions, deletions and synonym replacements, as well as combining or splitting multiple sentences to change their structure).

The task of matching the longest common subsequence is an important one in the many sub-domains of computer science. Its most naive implementation was deemed to have a time complexity of $O(m_1 * m_2)$ (where m_1 and m_2 are the numbers of concepts in compared documents). The question whether it is possible to achieve significantly better results was first raised by Knuth in [13].

One of the most important implementations of the search for the longest common subsequence is to be found in [19], which features time complexity $O((m_1 * m_2)/log(m_2))$. This work presents an application of Smith-Waterman algorithm for matching the longest common subsequence in textual data. This is a top achievement of algorithms that do not operate with text frames and their hashes. Other works such as [15], [21] or [23] prove that better efficiency is yielded rather by careful engineering strategies than a fundamental change in time complexity. All of the above cited works use algorithms, which time complexity is near quadratic which results in drastic drop of efficiency when dealing with documents of considerable length.

It was first observed in [22] that the introduction of a special structure based on the hashing technique, which was known later as shingling or chunks (a continuous sequence of fixed-length tokens in a document) can substantially improve the efficiency of deciding about the level of similarity of two documents by observing the number of common shinglings. This technique was introduced to detect near duplicate web pages. The following works such as [3], [1] introduce further extensions to the original idea.

Charikar [12] proposed a locality sensitive hashing scheme for comparing documents. A number of works represented by publications such as [4] or [2] have provided plausible methods to further boost the measuring of similarities between entities. Later, Henzinger [16] combined the algorithms of Broder et al. and Charikar to improve overall precision and recall. Recently Zhang [28] suggested a new algorithm based on sequence matching, which determines the location of duplicated parts in documents. Algorithms based on shingling are commonly utilized to identify duplicates or near-duplicates because of their ability to perform clustering tasks in linear computational complexity[24][17].

The important distinction between those given above and the $SHAPD$ (version 1 and version 2) is the emphasis on a sentence as the basic structure for comparison of documents and a starting point of determining a longest common subsequence. Thanks to such an assumption, SHAPD (version 1 and version 2) provides better results in terms of time needed to compute the results. Moreover, its functioning does not end at the stage of establishing that two or more documents overlap. It readily delivers data on which sequences overlap, the length of the overlapping and it does so even when the sequences are locally discontinued.

The capability to perform these makes it a method that can be naturally chosen in plagiarism detection, because such situations are common when attempt to hide plagiarism. In addition, it implements the construction of hashes representing the sentence in an additive manner, thus word order is not an issue while comparing documents.

$W - shingling$ algorithm runs significantly slower when the task is to give a length of a long common subsequence. Due to the fixed frame orientation, when performing such operation $w - shingling$ behaves in a fashion similar to the Smith-Waterman algorithm resulting in a significant drop of efficiency.

The importance of plagiarism detection is recognized in many publications. One might argue that, it is an essential task in times, where access to information is nearly unrestricted and a culture for sharing without attribution is a recognized problem (see [25] and [5]). Yet, as this work presents a special case of an algorithm for a longest common subsequence that can be used in other applications.

III. The SHAPD2 algorithm

Hashing is a commonly used technique in Natural Language Processing tasks used in order to achieve faster word retrieval. In plagiarism detection it is crucial, however, to identify and match longer common word sequences (with special focus on sentences).

SHAPD2 focuses on whole sentence sequences. A natural way of splitting a text document is to divide it into sentences and it can be assumed that documents containing the same sequences also contain the same sentences.

However, in text documents, there are situations in which there are cases when there are extremely long passages of text without a full-stop mark (such as different types of enumerations, tables, listings, etc.). Some sort of strategy needs to be devised for such cases, i.e. how to split portions of text, which are longer than the reasonable length of a sentence in a natural language.

$SHAPD2$ utilizes a brand new mechanism to organizing the hash-index as well as to searching through the index. It uses additional data structures such as correspondence array to aid in the process.

As introduced before, there are two corpora of documents comprise the algorithm's input: a corpus of source documents (originals) $D = \{d_1, d_2, ..., d_n\}$, and a corpus of suspicious documents to be verified regarding possible plagiaries, $P = \{p_1, p_2, ..., p_r\}$.

Before applying algorithm for there is necessary to carry out text-refinement process what is standard procedure in NLP/IR task (starting from unstructured text document input to a structure containing stacked sequentially descriptors of concepts found in the input document). Action that make up the process of text-refinement in documents starts from extracting lexical units (tokenization), and further text-refinement operations are: elimination of the words without semantic importance from the so-called information stop-list, the identification of multiword concepts (when phrase of several words create one concept), bringing concepts to the main form by lemmatization (for Polish documents) or stemming (for English documents using a popular Porter stemmer [26]). It is particularly difficult task for highly flexible languages, such as Polish, Russian or French (multiple noun declination forms and verb conjugation forms). In lemmatization procedure there is used Ispell dictionary for Polish documents and finite state automaton (FSA). The goal of both stemming and lemmatization is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form.

Synonyms need to be represented with the same concept descriptors using lexical relationships of synonymy from semantic network. It allows correct similarity analysis and also increases classification algorithms efficiency without loss in comparison quality [18].

Abstracting process faces another problem here, which is polysemy. One word can represent multiple meanings, so the apparent similarity need to be eliminated. It is done by concept disambiguation, which identifies word meaning depending on its context, is important to ensure that no irrelevant documents will be returned in response to a query [20].

The final effect of refinement procedure is the structure of documents containing ordered descriptors of concepts derived from the input document. This structure can be stored as an abstract (data for creating index) of the document, and then use during phase 2 (comparing documents).

Then, all documents need to be split into text frames of comparable length – preferably sentences, or in the case of

TABLE I. SAMPLE KEY DETERMINATION IN $SHAPD2$ ALGORITHM

davies_scott_1998_1.pdf
Applying Online Search Techniques to Reinforcement Learning Scott Davies Andrew Y. Ng Andrew Moore
In reinforcement learning it is frequently necessary to resort to an approximation to the true optimal value function. Here we investigate the benefits of online search in such cases. We examine local searches, where the agent performs a nite-depth lookahead search, and global searches, where the agent performs a search for a trajectory all the way from the current state to a goal state.

hash value	sentence/frame
4176335	reinforcement learning frequently necessary resort approximation true optimal value function
1699726	investigate benefits online search cases
2842476	examine local searches agent performs nite depth
2710940	lookahead search global searches agent performs search
2448654	trajectory way current state goal state

dayan-92.pdf
The Convergence of TD(X) for General X PETER DAYAN Machine Learning, 8, 341-362 (1992), Kluwer Academic Publishers, Boston, 1992
The methods of temporal differences (TD), first defined as such by Sutton (1984; 1988), fall into this simpler category. Given some parametric way of predicting the expected values of states, they alter the parameters to reduce the inconsistency between the estimate from one state and the estimates from the next state or states.

hash value	sentence/frame
2906878	methods temporal differences td defined sutton fall simpler category
2496872	given parametric way predicting expected values states alter
2339613	parameters reduce inconsistency estimate state estimates state states

longer sentences – shorter phrases. A coefficient α is a user-defined value, which allows to set the expected number of frames that a longer sentence is split into. The coefficient ranges from 6 to 12 concepts. The new procedure of uniform fragmentation is described in listing of Algorithm 1.

Algorithm 1 Phase 1. Splitting text into comparable frames

```
$f := round_up(l/\alpha)$
while $f > 0$ do
    $a := round_up(d/f)$
    $\bar{c} := getConceptsFromSentence(s, a)$
    $calculateHash(\bar{c})$
    $l := l - a$
    $f := f - 1$
end while
$l$ - sentence $s$ length
$\alpha$ - alpha coefficient
$f$ – number of frames to split a longer sentence into
$a$ – current frame length
$\bar{c}$ – vector of frame concepts
```

The first version of the SHAPD algorithm was able to compare a suspicious document with exactly one document from the corpus P. The SHAPD2 algorithm is able to compare a suspicious document with the entire corpus of source documents. For all documents d_i from corpus P (containing suspicious documents), the correspondence array CL and maxima array TM are cleared. For each frame, set of tuples is retrieved from index table T. If there are any entries existing, it is then checked whether they point to the same source document and to the previous frame. If the condition is true, local correspondence maximum is increased by one. Otherwise, the local maximum is decreased.

After all of the frames are checked, table TM storing the correspondence maxima are searched for records whose correspondence maxima are greater than a threshold set e (the number of matching frames to be reported as a potential plagiarism). Frame and document number are returned in these cases.

Sample outputs from a sentence splitting and hash calculation in Phase 1 are shown in Table I. As a result, every document from the original corpus, as well as all suspicious documents, is represented by index as a list of sentence hashes. The first version of the SHAPD algorithm was able to compare a suspicious document with exactly one document from the corpus P. The SHAPD2 algorithm is able to compare a suspicious document with the entire corpus of source documents in single pass.

In the next step, a hash table T is created for all documents from corpus D, where for each key the following tuple of values is stored: $T[k_{i,j}] =< i, j >$, (document number, frame number) (see Figure 1).

For phase 2, a correspondence list CL is declared, with elements of the following structure: n_d – document number, m_l – local maximum, and n_l – frame number for local sequence match.

Another data structure is the maxima array TM for all r documents in corpus P, which contains records with structure as follows: m_g – global maximum, n_g – frame number with global sequence match.

Phase 2 is performed sequentially for all documents from corpus P. Its logic is listed in Algorithm 2.

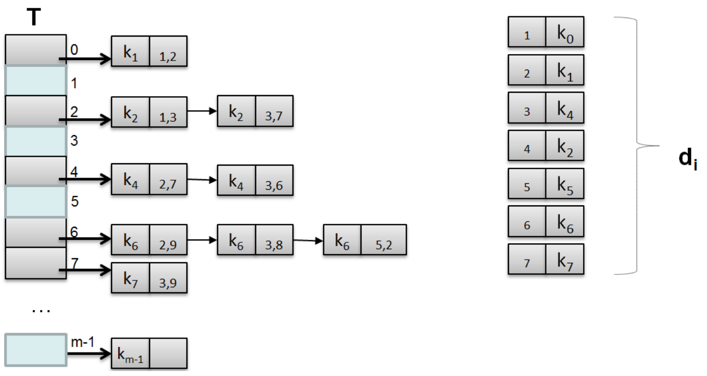

Fig. 1. Hash table T indexing a corpus D of source documents and hashes from suspicious document $d_i \in P$

IV. EXPERIMENT

In order to evaluate the algorithm's efficiency, a series of experiments were carried out. The most widely used test collection Reuters-21578 [27] for text categorization has been used as a source of the testing data. The Reuters-21578 corpus consists of multiple sets of annotated documents. A collection of source documents (originals) as well as suspicious documents (including potential plagiarism cases) has been derived from the Reuters corpus.

Algorithm 2 Phase 2

```
{Build hash table T}
for $d_i \in D$ do
    for $frame_j \in d_i$ do
        $addHashToIndex(T)$
    end for
end for{Find longest common frame sequences}
for $d_i \in P$ do
    $clear(CL)$
    $clear(TM)$
    for $frame_j \in d_i$ do
        if $existsT(k_{i,j})$ then
            if $T(k_{i,j}).i = i \wedge T(k_{i,j}).j = j - 1$ then
                $CL(i).m_l + +$
                $update(TM)$
            end if
        else
            $CL(i).m_l - -$
        end if
    end for
    for $tm \in TM$ do
        if $tm.m_g > e$ then
            $return(tm.n_g)$
        end if
    end for
end for
```

TABLE II. Details of comparison (presented in Figure 1) between a suspicious document d_i and a corpus D represented by hash table T

	$k_i \in T$	CL table	TM table
1	$k_0 \notin T$		
2	$k_1 \in T$	$\{m_l = 1, n_d = 1, n_l = 2\}$	$TM_1 = \{m_g = 1, n_g = 2\}$
3	$k_4 \in T$	$\{m_l = 1, n_d = 2, n_l = 7\}$	$TM_2 = \{m_g = 1, n_g = 7\}$
		$\{m_l = 1, n_d = 3, n_l = 6\}$	$TM_3 = \{m_g = 1, n_g = 6\}$
4	$k_2 \in T$	$\{m_l = 1, n_d = 1, n_l = 3\}$	
		$\{m_l = 2, n_d = 3, n_l = 7\}$	$TM_3 = \{m_g = 2, n_g = 7\}$
5	$k_5 \notin T$	$\{m_l = 1, n_d = 3, n_l = 7\}$	
6	$k_6 \in T$	$\{m_l = 1, n_d = 2, n_l = 9\}$	
		$\{m_l = 2, n_d = 3, n_l = 8\}$	
		$\{m_l = 1, n_d = 5, n_l = 2\}$	$TM_5 = \{m_g = 1, n_g = 2\}$
7	$k_7 \in T$	$\{m_l = 3, n_d = 3, n_l = 9\}$	$TM_3 = \{m_g = 3, n_g = 9\}$

k_j – hash key of frame j from document d_i

A set of 3,000 original documents was used as source set, and several sets including 1,000 to 6,000 suspicious documents were used as a set of suspicious documents. The sets were compared using two algorithms: w-shingling and SHAPD2. All tests were carried out on one computing platform, a stock laptop computer with an 8-core processor (four cores in hyper-threading mode), clocked at 2.0 GHz.

The basic results of efficiency test are as follows. A comparison of one suspicious document to a set of 3,000 originals (containing about 3060 words on average) takes 7.13 milliseconds. As many as 420,700 document-to-document comparisons were achieved in 1-second intervals.

As the hash-table remains unchanged after Phase 1, it is possible to run further processing in parallel threads, because of a sequential way of comparing the indiviual suspicious documents. In effect, further acceleration has been achieved, resulting in single comparison time of 3.45 milliseconds instead of 7.13 milliseconds.

TABLE III. Processing time [ms] for comparing n suspicious documents with a corpus of 3,000 original documents

n	w-shingling	SHAPD2
1000	5680	4608
1500	6654	5114
2000	8581	5820
2500	9478	6374
3000	11967	7125
3500	14864	7213
4000	16899	7527
4500	20242	7818
5000	23200	8437
5500	33955	8656
6000	50586	8742

The resulting times of tests' execution are presented in Tables III and IV and visualized on Figure 2. One can observe that the efficiency turns out to be better by an order of magnitude from $w - shingling$. This enables a new area of applications, including plagiarism detection using much larger source document corpus, as well as making document comparisons against a medium-size corpus possible in nearly on-line time.

If two sentences differ only in one concept, the hash keys created for them will be different, even when those concepts are synonyms. In such situation the last operation in text refinement procedure should be a generalization of concepts using semantic compression. Moreover, the algorithm uses the same techniques whose high effectiveness has already been proven [10] in plagiarism detection employing semantic compression[7], as well as their strong resilience to false-positive examples of plagiarism[9], which may be an issue in cases when competitive algorithms are used.

The Clough & Stevenson Corpus of Plagiarised Short Answers[14] was used for the benchmark. It serves the purpose of plagiarism detection as it is built from an initial set of documents as the base, which was then altered by the participants in a number of ways so that everyone could measure how well their approach to plagiarism detection works.

One might address the issue of the common ancestor as a document that is cited the most in a number of documents. Thanks to $SHAPD2$, one can easly handle it with no additional modifications except for the length of text frame used in given application[10].

It is even more interesting to apply $SHAPD2$ in conjunction with semantic compression in order to handle situations in which author has not cited sources, yet used formulations that can be traced to some actual document.

V. Conclusions

To summarize, there should be emphasizee the following properties of the SHAPD2 algorithm:

- The algorithm developed has a very low computational complexity – evaluated to linearithmic, proving to be extremely efficient in a task of finding long common

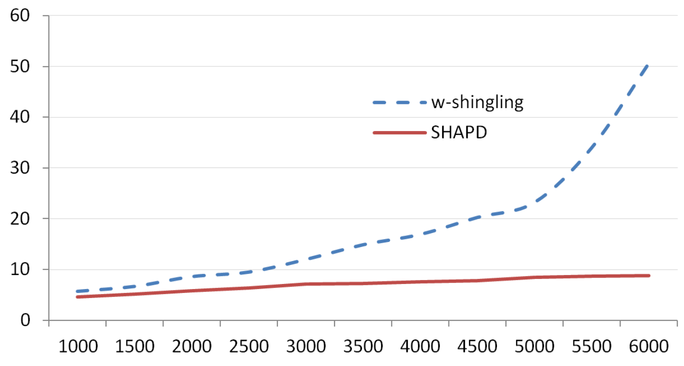

Fig. 2. Processing times (ms) of comparing n suspicious documents with a corpus of 3,000 documents using $w - shingling$ and $SHAPD2$ algorithms

TABLE IV. PROCESSING TIME [MS] FOR COMPARING 3,000 SUSPICIOUS DOCUMENTS WITH A CORPUS OF n ORIGINAL DOCUMENTS

n	SHAPD2
1000	3142
1500	4141
2000	5239
2500	5704
3000	7137
3500	9194
4000	9722
4500	10563
5000	11678
5500	15270
6000	22182

sequences in compared documents. This was confirmed on the experiments with Reuters-21578 corpus.

- $SHAPD2$ is resilient to fluctuating word order in sentences. It is especially important in use cases in languages with a highly flexible syntax (e.g. Polish, which allows multiple correct word sequences, although having a SVO-based (Subject-Verb-Object-based) syntax, like English).
- The algorithm is resilient to small sentence inclusions or deletions (as the Smith&Waterman algorithm is), which is an important feature in plagiarism detection, as it is a common strategy of slight modifications when committing plagiaries.
- The designed $SHAPD2$ algorithm employing semantic compression is highly efficient in plagiarism detection and is strongly resilient to false-positive examples of plagiarism, which may be an issue in cases when competitive algorithms are used.
- Utilization of NLP techniques, such as term identification and disambiguation, semantic compression, would surly improve the effectiveness of plagiarism detection, which is subject to further developments.

REFERENCES

[1] A. Andoni and P. Indyk, ”Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions”, *Commun. ACM*, 51(1), January 2008, pp. 117–122.

[2] O. A. Hamid, B. Behzadi, S. Christoph, and M. Henzinger, ”Detecting the origin of text segments efficiently”, Proceedings of the 18th International Conference on World wide web WWW 09, vol. 7(3), 2009, pp. 61–70.

[3] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig, ”Syntactic clustering of the web”, Computer Networks and ISDN Systems, vol. 29. n. 8–13, September 1997, pp. 1157–1166.

[4] A. Z. Broder, ”Identifying and filtering near-duplicate documents”, In: COM ’00: Proceedings of the 11th Annual Symposium on Combinatorial Pattern Matching, London, UK. Springer Verlag, Berlin Heidelberg, 2000, pp. 1–10.

[5] S. Burrows S., S. M. Tahaghoghi, and J. Zobel, ”Efficient plagiarism detection for large code repositories”, Software Practice and Experience, vol. 37, 2007, pp. 151–175.

[6] Ceglarek Dariusz, Haniewicz Konstanty, and Rutkowski Wojciech, ”Semantically Enhanced Intellectual Property Protection System – SEIPro2S, Proceedings of the 1st International Conference on Computational Collective Intelligence, Springer Verlag, Berlin Heidelberg, 2009, pp. 449–59,

[7] D. Ceglarek, K. Haniewicz, and W. Rutkowski, ”Semantic compression for specialised information retrieval systems”, In: N. T. Nguyen, R. Katarzyniak, and S. M. Chen, (eds.), vol 283 of Studies in Computational Intelligence - Advances in Intelligent Information and Database Systems, Springer Verlag, Berlin Heidelberg, 2010, pp. 111–121.

[8] D. Ceglarek and K. Haniewicz, ”Fast plagiarism detection by sentence hashing”, In: L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L. A. Zadeh, and J. M. Zurada (eds.), vol. 7268 of Lecture Notes in Computer Science - Artificial Intelligence and Soft Computing, Springer Verlag, Berlin Heidelberg, 2012, pp. 30–37.

[9] D. Ceglarek, K. Haniewicz, and W. Rutkowski, ”Robust Plagiary Detection Using Semantic Compression Augmented SHAPD”, In: N. T. Nguen and R. Katarzyniak (eds.), vol. 7653 of Lecture Notes in Artificial Intelligence - Computational Collective Intelligence - Technologies and Applications, Springer Verlag, Berlin Heidelberg, 2012, pp. 308–317.

[10] D. Ceglarek and K. Haniewicz, ”Detection of the Most Influential Documents”, In: M. Pechenizkiy and M. Wojciechowski (editors), vol 185 of Advances in Intelligent Systems and Computing - New Trends in Databases and Information Systems, Springer Verlag, Berlin Heidelberg, 2013, pp.49–59.

[11] D. Ceglarek, ”Architecture of the Semantically Enhanced Intellectual Property Protection System”, Advances in Intelligent and Soft Computing - Computer Recognition System vol. 5, pp. 12, Springer Verlag, Berlin Heidelberg, 2013.

[12] M S. Charikar, ”Similarity estimation techniques from rounding algorithms”, Proceedings of the thiry-fourth annual ACM symposium on Theory of computing, STOC ’02, ACM, 2002, pp. 380–388.

[13] V. Chvatal, D. A. Klarner, and D. E. Knuth, ”Selected combinatorial research problems”, Technical report, Stanford, CA, USA, 1972.

[14] P. Clough and M. Stevenson, ”A Corpus of Plagiarised Short Answers”, 2009. [http://ir.shef.ac.uk/cloughie/resources/plagiarism_corpus.html, accessed 2 April 2012].

[15] C. Grozea, Ch. Gehl, and M. Popescu, ”Encoplot : Pairwise sequence matching in linear time applied to plagiarism detection”, Time, 2009, pp. 10–18.

[16] M. Henzinger, ”Finding near-duplicate web pages: a large-scale evaluation of algorithms”, In: SIGIR ’06: Proceedings of the 29th annual International ACM SIGIR Conference on Research and development in information retrieval, New York, NY, USA, ACM, 2006, pp. 284–291.

[17] Timothy C. Hoad , Justin Zobel, ”Methods for identifying versioned and plagiarized documents”, Journal of the American Society for Information Science and Technology, vol. 54 n. 3, 2003, pp. 203–215.

[18] A. Hotho, S. Staab, and G. Stumme, ”Explaining Text Clustering Results using Semantic Structures”, Principles of Data Mining and Knowledge Discovery, September 2003, pp. 217-228.

[19] R. W. Irving, ”Plagiarism and collusion detection using the Smith-Waterman algorithm”, Technical report, University of Glasgow, Department of Computing Science, 2004.

[20] R. Krovetz and W. B. Croft, ”Lexical Ambiguity and Information Retrieval”, 1992.

[21] R. Lukashenko, V. Graudina, and J. Grundspenkis, ”Computer-based plagiarism detection methods and tools: an overview”, Proceedings of the 2007 International Conference on Computer systems and technologies, CompSysTech ’07, New York, NY, USA, ACM, 2007, pp. 40:1–40:6.

[22] U. Manber, ”Finding similar files in a large file system”, Proceedings of the USENIX Winter 1994 Technical Conference on USENIX Winter 1994 Technical Conference, WTEC’94, Berkeley, CA, USA. USENIX Association, 1994, pp. 2–2.

[23] M. Mozgovoy, S. Karakovskiy, and V. Klyuev, ”Fast and reliable plagiarism detection system”, In: Frontiers In Education Conference – Global Engineering: Knowledge Without Borders, Opportunities Without Passports, 2007. FIE ’07, 37th Annual, 2007, pp. S4H–11 –S4H–14.

[24] R. Nock and F. Nielsen, ”On weighting clustering”, Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 28. n. 8, 2006, pp. 1223–1235.

[25] T. Ota and S. Masuyama, ”Automatic plagiarism detection among term papers”, In *Proceedings of the 3rd International Universal Communication Symposium*, IUCS ’09, New York, NY, USA, ACM, 2009, pp. 395–399.

[26] M. F. Porter, ”An algorithm for suffix stripping”, Program 14 (3), 1980, pp. 130-137.

[27] Reuters-21578 Corpus, Carnegie Group, Inc. and Reuters, Ltd, 2004 [http://www.daviddlewis.com/resources/testcollections/reuters21578, access 10 April 2012]

[28] Q. Zhang, Y. Zhang, H. Yu, and X. Huang, ”Efficient partial-duplicate detection based on sequence matching”, SIGIR ’10: Proceeding of the 33rd international ACM SIGIR Conference on Research and development in information retrieval, New York, NY, USA, ACM, 2010, pp. 675–682.