Plagiarism Detection Through Multilevel Text Comparison

Manuel Zini, Marco Fabbri, Massimo Moneglia, Alessandro Panunzi
Università di Firenze
Italian Department
mlzini@yahoo.com, marcofab30@yahoo.it, moneglia@unifi.it, alessandro.panunzi@unifi.it

Abstract

The paper presents the implementation of a tool for plagiarism detection developed within the AXMEDIS project. The algorithm leverages the plagiarist behaviour, which is modeled as a combination of 3 basical actions: insertion, deletion, substitution. We recognize that this behaviour may occur at various level of the document structure: the plagiarist may insert, delete or substitute a word, period or a paragraph. The procedure consists in two main steps: document structure extraction and plagiarism function calculation. We propose a recursive plagiarism evaluation function to be evaluated at each level of the document structure which is based on the Levenshtein edit distance. We also propose a method that will eliminate unnecessary chunks comparison, avoiding similarity calculation of chunks which do not share enough 4-grams. We describe the similarity algorithm and discuss some implementation issues and future work.

1. Introduction

Plagiarism is a growing problem and has recently received a lot of attention. The increase in availability of material in digital form has made plagiarism much easier. However, the use of digital media also means that there is greater opportunity to trace plagiarism by means of automation in form of software tools. Automated plagiarism detection as a subject has not yet achieved the same degree of scientific maturity that other subjects have already achieved, but a growing number of publications, websites and recently available products on this matter [2][3][4][12][13] indicates that people have started to recognise and acknowledge the existence of a recent problem which is yet awaiting its systematic solution.

There are several approaches to automatically identify plagiarism in different types of documents. The SCAM tool developed by Narayanan Shivakumar [9] is based on building unions of word sets and counting domain-specific keywords in them. Plagiarism is then revealed via unexpected or otherwise suspicious occurrences of such keywords. SCAM has reportedly found serious cases of science plagiarism in the DBWORLD domain of research literature about databases. Thousands of DBWORLD entries had been scanned for that purpose.

In some works, plagiarism detection has been regarded as a special case of duplicate document detection, which is a both necessary and difficult task in the management of large scale and very large scale databases (possibly multi-media databases). A variety of data mining methods and text-based techniques for such purposes have been described and investigated [5].

One approach to copy detection is to incorporate a watermark into a document [8][7]. This watermark identifies the original user who requests the document. If the document is electronically copied, i.e. the document is possessed by a person other than the original user, a violation has occurred, and the watermark can witness the culprit. The major weakness of the watermark scheme is that the watermark itself can be destroyed by processing the document through a lossy compression operation like JPEG. In addition, detection of partial copying is not possible. Watermarking is a kind of passive copy detection approach.

Comparing whole document checksums is simple and suffices for reliably detecting exact copies; however, detecting partial copies is subtler; in some works [10] an approach based on multiple fingerprints evaluation is used to detect partial copies. These techniques mostly rely on the use of k-grams, i.e. contiguous sub-strings of characters with length k. The process requires to divide a document into k-grams, and extract an hash value from each one [14][15]. The result of this process is a fingerprint that represents the document in each of its sub-parts of length k, further exploited for comparison. Such a procedure, however, does not take into account the behavioural pattern of the plagiarist. In [6] the edit distance is introduced as a similarity metric between chunks of text.

In our work, which will be distributed as a security plugin within the AXMEDIS framework, we developed

Proceedings of the Second International Conference on Automated
Production of Cross Media Content for Multi-Channel Distribution (AXMEDIS'06)
0-7695-2625-X/06 $20.00 © 2006 IEEE

a model of the plagiarist behaviour and observe that this behaviour applies at any level of the document structure (word, period, paragraph). In the AXMEDIS Framework this plug-in function can be useful to content producers and distributors partners. If the user suspects that some text documents, maybe found on the net, could have been produced with a "cut&paste" processing of his original and copyright protected content, he can detect in an automatic way which of his documents have been more likely exploited. Just as an example the user could create a rule for AXMEDIS RuleEditor which processes a set of possessed text documents against a set of similar documents potentially derived from a plagiarism action, and get a list of the more relevant.

In section 3 we describe such a model defining the operations. In section 4.4 we define a recursive plagiarism function which takes into account the modeled behavioural pattern at any structural level. The function gives a value between 0 and 1 which represent the degree of plagiarism between the compared documents. In section 5 we sketch the designed algorithm. Since the computation of this function is time consuming we also propose a method to avoid unnecessary comparisons based on the idea that two copied chunks of a document should share at least a percentage of 4-grams word patterns. If they do not the plagiarism function will not be evaluated. In section 6 and 7 we discuss parameters identification and testing methodology.

2. Preliminary definitions

Let D_i and D_j be two documents to be compared. Let T be the structure tree of the document, where each child node represents a word. Let T_i represent the structure tree of document D_i. We number the tree levels starting from the child level, so level 0 nodes represent words, level 1 nodes represents periods, level 2 nodes represents paragraphs. We refer to words, periods, and paragraphs generally as 'chunks'. We define c_i^L as a chunk of text representing a node of level L of the document tree T_i. Moreover we define:

$$\lambda(c) := \text{the number of contained sub-chunks} \quad (1)$$

$$|c| := \text{the character length of chunk c} \quad (2)$$

$$\sigma(c^L) \text{ is the set of sub-chunks of level L-1 in } c^L. \quad (3)$$

3. The plagiarist behaviour

From our observations follows that the plagiarist behaviour consists of some prototypical actions performed on the original text to obtain a new text. This prototypical actions are: insertion, deletion, substitution. We observe also that these actions can be performed at any level of the document structure, the plagiarist could for instance delete one paragraph from the document, insert one period in a paragraph, change one word in a period and so on.

Let $s1$ be a sub-sequence of chunks belonging to the same level L extracted from the original document. Let $s2$ be the sequence of chunks produced by the plagiarist starting from $s1$.
Let a,b,c be chunks of the same level L. We observe that the plagiarist may perform the following simple actions:

- Insertion
  The plagiarist may start from the original sequence $s1$ and insert a chunk.
  $s1 = ab \rightarrow s2 = acb$
- Deletion
  The plagiarist may start from the original sequence of chunks $s1$ and delete a chunk.
  $s1 = acb \rightarrow s2 = ab$
- Change
  The plagiarist may start from the original sequence of chunks $s1$ and substitute a chunk.
  $S1 = acb \rightarrow s2 = adb$

4. Document comparison and plagiarism function evaluation

4.1 Document structure extraction

The pre-processing module tokenizes the documents and identifies periods and paragraphs. We refer to words, periods, and paragraphs generally as 'chunks'. Words are chunks of level 0, periods are chunks of level 1 and so on. A document structure tree is created, where each node identifies a chunk, and each child of the node represents a contained chunk. Chunks will then be compared through the evaluation of the plagiarism function. Comparison will start at the highest level of the tree, only chunks at the same level will be compared.

4.2 The edit distance

We choose the Levenshtein distance as a basis for our plagiarism function because it takes into account the insertion, deletion, substitution behaviour of the plagiarist. The algorithm we use for distance calculation is described in [11]. The algorithm yields the minimum distance between two strings defined in terms of editing primitives. The primitives considered are, as stated earlier, insertion, deletion, and substitution. Given two character strings, the idea is to find the smallest set of primitives that applied to one string

produces the other. The number of primitives in the set is the string distance. The algorithm applies unmodified to word token comparison, we adapt it to allow for any kind of chunk sequence to be compared: given two chunks they will be considered equivalent for the edit distance calculation if their plagiarism function is greater than a threshold, τ_L, which can be set differently for each level.

4.3 Multi level comparison

We observed that the plagiarist behaviour as described in section 3 applies at any level of the document structure. The plagiarist performs actions on the original document that modify the structure of the document itself. To leverage this observed behaviour we have designed a recursive plagiarism function that compares documents at level L taking into account structural differences at level L and a weighted average of the same function evaluated for each chunk combination at level L-1. Since the plagiarism function evaluation is time consuming we have defined a criterion to avoid unnecessary comparison between chunks which is $O(n)$ in time.

4.4 Plagiarism function definition

4.4.1 Structural similarity function

Let D_i and D_j be two documents. Let c_i^L and c_j^L be two level L chunks such that $c_i^L \in D_i$ and $c_j^L \in D_j$.

We define the structural similarity as:

$$\zeta(c_i^L, c_j^L) = 1 - \frac{ed(c_i^L, c_j^L)}{\max(\lambda(c_i^L), \lambda(c_j^L))} \quad (4)$$

where $ed()$ is the edit distance between the two chunks as defined in section 4.2 and $\lambda(c)$ is defined as in (1).

4.4.2 Chunk similarity function

Let c_i^L and c_j^L be two level L chunks such that $c_i^L \in D_i$ and $c_j^L \in D_j$,

We define chunk similarity of level L:

$$\xi(c_i^L, c_j^L) = \frac{\sum w(c_i^{L-1}, c_j^{L-1}) P(c_i^{L-1}, c_j^{L-1})}{\sum w(c_i^{L-1}, c_j^{L-1})} \quad (5)$$

where $(c_i^{L-1}, c_j^{L-1}) \in \sigma(c_i^L) \times \sigma(c_j^L)$,

$w(c_i^{L-1}, c_j^{L-1}) = \max(\frac{|c_i^{L-1}|}{|c_i^L|}, \frac{|c_j^{L-1}|}{|c_j^L|})$

and σ is defined as in (3)

4.4.3 Plagiarism function

Having defined the structural and chunk similarity function we are now able to define the plagiarism function as:

$$P(c_i^L, c_j^L) = \begin{cases} \frac{\alpha_L \xi(c_i^L, c_j^L) + \beta_L \zeta(c_i^L, c_j^L)}{\alpha_L + \beta_L} & if L > 0 \\ \zeta(c_i^L, c_j^L) & if L = 0 \end{cases} \quad (6)$$

where α_L and β_L are adjustable weight parameters.

5. The proposed algorithm

The algorithm we propose makes the assumption that it will receive two textual documents which have already been evaluated as being on the same subject, since no comparison is needed between documents which are on different subjects. The subjects identification can be performed using the keyword extraction plugin described in [1], two documents will be considered on the same subject if they share enough keywords. We also assume that newline characters represent paragraphs. The first step that the algorithm performs is document tokenization, then it builds a document structure tree as described in section 2. While building the structure tree we also extract each 4-gram of the document marking to which node of the tree it belongs. These data will be used for a rough evaluation of similarity to be performed in linear time to avoid unnecessary plagiarism function evaluations. After the tree has been built the evaluation of the plagiarism function starts from level 3 (document level). For each calculation of the function we perform first a comparison of the extracted 4-grams, comparing the percentage of matches against a threshold g_L which can be different for each level. If the percentage of matching 4-grams is lower than g_L the value of the plagiarism function is set to 0. As we stated earlier this comparison can be done in linear time and avoids the heavy calculations required to evaluate P. Moreover when we compute the value of P for level 0 we are comparing single words, and the value of P is set to be the value of the string edit distance between words, as stated in the function definition (6).

The algorithm terminates returning a boolean value S. S is true if the plagiarism evaluation function (6) is greater than a threshold t:

$$S(D_i, D_j) == \begin{cases} true & if P(D_i, D_j) > t \\ false & if P(D_i, D_j) \leq t \end{cases} \quad (7)$$

if S is true it means that one of the two documents is possibly derived from the other, applying modifications that are typical of the plagiarist behaviour, and as such, is suspect of having been plagiarized. It ought to be noted that the proposed algorithm is not able nor meant to detect every possi-

ble kind of plagiarism. Plagiarism can be performed in several ways, for instance a plagiarist may change completely the written representation of a document and just keep ideas and concepts. Our algorithm is a plagiarist behaviour detector as defined in section 3. It has to be noted, by the way, that the forementioned behaviour, which is eased by digital tools like cut & paste, is likely to become really common, given the ever growing digital nature of documents.

6. Parameters Identification

The proposed algorithm has several parameters that have to be identified in order to maximize precision and recall on a training set. For each level L, α_L and β_L have to be identified together with τ_L. Moreover the threshold t used in (7) must be identified. We will identify parameters choosing those values that maximize precision and recall on an automatically generated training set. In order to build the needed set of documents we developed an automatic document generator specifically designed to perform modifications emulating the behaviour of a plagiarist. The document generator starts from a seed document and modifies the document simulating the plagiarist behaviour as described in section 3. The intensity of the generated variations is random but never exceeds a threshold. This threshold is independently adjustable for each type of modification.

In order to identify the parameters we will proceed as follow:

Let $\overline{a}, \overline{b}$ be two seed documents, taken from the real world and on the same subject but clearly independent from each other.

- Let A, B be the set of documents derived respectively from $\overline{a}, \overline{b}$ applying the algorithm for plagiarist behaviour simulation.
- Let S be the boolean function for plagiarism detection as defined in (7).
- Let TP be the set of documents a ϵ A such that $S(a, \overline{a}) = true$
- Let FN be the set of documents a ϵ A such that $S(a, \overline{a}) = false$
- Let TN be the set of documents b ϵ B such that $S(b, \overline{a}) = false$
- Let FP be the set of documents b ϵ B such that $S(b, \overline{a}) = true$

We define precision PR and recall RC as:

$$PR = |TP|/(|FP| + |TP|) \quad (8)$$

$$RC = |TP|/(|TP| + |FN|) \quad (9)$$

After having generated the training set we will identify a set of parameters values that maximize PR and RC. A detailed description of the results will be reported in a following paper.

7. Testing method

In order to test the algorithm with the identified parameters we developed an experimental methodology to build a corpus of plagiarized documents. We start from several sets of real world documents. Each set contains documents covering the same subject. Documents in a set are not derived from each other and contain at least one clearly distinctive concept. We select a group of persons informed on the subjects they are going to deal with but not experts. For each person a subject is assigned. The set of documents on the same subject is divided in two subsets, say A and B. We ask the selected person to create a new document on the subject starting only from documents in set A. The assigned goal is to generate a new document o on the subject which is as original as possible. The task has to be performed in a given time limit which is quite tight compared to the document length. The produced document should be meaningful, grammatically correct and not smaller than the average size of the documents. For each subject a reviewer is then asked to assess the generated documents. For each generated document o the reviewer should list which documents in set A he reckons have been used to write it, defining a subset of A that we call $\overline{A}(o)$. The algorithm will then be tested against generated documents. Any comparison of a generated document o with a document of set $\overline{A}(o)$ should give positive result, any comparison with documents in set B should give a negative result. Recall RC and precision PR will then be calculated. The results of this test will be reported in a following paper.

8. Conclusions and future work

In this paper we have presented a method for the comparison of documents aimed at spotting plagiarism. We took into consideration a model of the behaviour of the plagiarist and observed that this behaviour applies at any level of the document structure. We defined a recursive function that takes into account a weighted inter-chunk similarity and a structural similarity based on the Levenshtein edit distance.

Proceedings of the Second International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution (AXMEDIS'06)
0-7695-2625-X/06 $20.00 © 2006 IEEE

The function gives a value between 0 and 1 which represents the degree of plagiarism between the compared documents. A threshold value that defines 'suspect documents' can be identified. Moreover we have designed a criterion aimed at reducing the computational cost of the overall algorithm avoiding unnecessary chunk comparisons. This criterion is based on the comparison of common 4-grams between chunks. Future work is foreseen in the experimental validation of the algorithm: a)Tuning of parameters and thresholds to achieve the best results will be performed according to section 6. b)Extensive testing will be performed according to the methodology expressed in section 7.

9. Acknowledgements

Thanks to all AXMEDIS project partners including the Expert User Group and all affiliated members, for their contributions, funding and collaborations, sorry if they have not been involved in the paper and have not been mentioned. We trust in their understanding. A specific acknowledgment to EC IST FP6 for partial funding of AXMEDIS project.

References

[1] Panunzi A., Moneglia M., Fabbri M. Keyword extraction in open-domain multilingual text resources. *Proceedings 1st International Conference on Automated Production of Cross Media Content for Multi-channel Distribution*, pages 253–256, 2005.

[2] Barret R. Are we ready for large scale use of plagiarism detection tools? *http://www.ics.ltsn.ac.uk/pub/conf2003/RuthBarrett.ppt*, 2003.

[3] Clough P. Plagiarism in natural and programming languages: An overview of current tools and technologies. *http://www.dcs.shef.ac.uk/cloughie/plagiarism/HTML Version/*, 2003.

[4] Culwin F. Practical free-text: Plagiarism investigation. *http://www.ics.ltsn.ac.uk/pub/conf2003/fintansTutorial.ppt*, 2003.

[5] Lopresti D. A comparison of text-based methods for detecting duplication in document image databases. *Proc of Document Recognition and Retrieval VII (IS and T SPIE Electronic Imaging) San Jose (USA)*, 3967:210–221, January 2000.

[6] Mandreoli P. F., Martoglia R. Un metodo per il riconoscimento di duplicati in collezioni di documenti. *Proceedings of the Eleventh Italian Symposium on Advanced Database Systems*, SEBD, 2003.

[7] Hiary K. H. Watermark: From paper texture to digital media. *Proceedings 1st International Conference on Automated Production of Cross Media Content for Multi-channel Distribution*, pages 261–264, 2005.

[8] Brassil N. M. J., Low S. Identification using both line and word shifting. technical report. *ATT Bell Labratories*, 1994.

[9] Shivakumar N. A real-life instance of plagiarism detection by scam. *http://www-db.stanford.edu/shiva/scam/plag.html*, 2003.

[10] Schleimer A. A. S., Wilkerson D.S. Winnowing: local algorithms for document fingerprinting. *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, 2003.

[11] Levenshtein V. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics-Doklady*, SEBD(10):707–710, 1996.

[12] EVE Plagiarism Detection System. *http://www.canexus.com/eve/index.shtmlKoala*

[13] Document Turnitin. *http://www.turnitin.com/static/home.html*

[14] Heintze, N. Scalable document fingerprinting. *In 1996 USENIX Workshop on Electronic Commerce*, pp.191-200, 1996.

[15] Koala Document Fingerprinting (KDF). *http://www-2.cs.cmu.edu/afs/cs/user/nch/www/koala-info.html*