Evaluating robustness for 'IPCRESS': Surrey's text alignment for plagiarism detection

Notebook for PAN at CLEF 2014

Lee Gillam, Scott Notley

Department of Computing, University of Surrey, UK
l.gillam@surrey.ac.uk

Abstract. This paper briefly describes the approach taken to the subtask of Text Alignment in the Plagiarism Detection track at PAN 14. We have now re-implemented our PAN12 approach in a consistent programmatic manner, courtesy of secured research funding. PAN 14 offers us the first opportunity to evaluate the performance/consistency of this re-implementation. We present results from this re-implementation with respect to various PAN collections, although it is important to note that our target is to be able to undertake plagiarism detection in such a way as would be impervious to a range of attempts to discover the content being matched against – a kind of privacy-preserving plagiarism detection.

1 Introduction

As reported in our PAN 13 notebook paper, having secured funding from the UK government-backed Technology Strategy Board for 18 months, the University of Surrey have been working on the Intellectual Property Protecting Cloud Services in Supply Chains (IPCRESS) project, a collaboration with Jaguar Land Rover and GeoLang Ltd. The IPCRESS project is focused on the difficulty of entrusting valuable Intellectual Property (IP) to third parties, through the Cloud, as is necessary to allow for the construction of components in the supply chain. The key innovation is the ability to track high-value IP without having to reveal that IP – so approaches need to avoid being reversible to text in clear. Such tracking is then suited to the tasks of (i) preventing IP leakage; (ii) detecting IP leakage or theft; and (iii) identifying retention beyond allowed review periods. The project builds from the proposed formulation of such a system in Cooke and Gillam 2011. This can be formulated as a kind of plagiarism detection, and hence the relevance of/to PAN, with a more challenging aim: to be able to generate reliable detections without access to the textual content – and so allowing for matches to be undertaken in public without exposing the content high-value documents that ought to be locked in secure electronic vaults. As such, only those with suitable access to the document in the vault should be able to verify the match.

In this paper, we briefly discuss the simplification of the code-base from our original submissions to the present and much more self-contained setup, and demonstrate the consistency of results obtained. We also hint at improvements in our treatment of obfuscation that are likely to become a focal point for future work also.

Section 2 provides a brief summary of results found with re-used software applied to PAN 2011, PAN 2012 and PAN 2013 datasets. Section 3 carries discussion of the IPCRESS re-implementation. Section 4 presents results of applying IPCRESS to the datsets for PAN 2012 and PAN 2013, and preliminary results found using initial obfuscation handling approaches. Section 5 comments on the PAN 2014 results and future work.

2 Previous PAN results

We have discussed in previous PAN efforts (e.g. Cooke, 2011) how our intention is to be able to find matching text without revealing the textual content. In PAN 11, the approach brought us 4th place, with PlagDet=0.2467329, Recall=0.1500480, Precision=0.7106536, Granularity=1.0058894. In 2012, we showed good granularity, with high recall and precision for non-obfuscated text, but not such great recall in the face of obfuscation (see Table below).

Test	Plagdet Score	Recall	Precision	Granularity
02_no_obfuscation	**0.92530**	**0.90449**	**0.94709**	**1.0**
03_artificial_low	0.09837	0.05374	0.93852	1.04688
04_artificial_high	0.01508	0.00867	0.96822	1.20313
06_simulated_paraphrase	0.11229	0.05956	0.97960	1.0

In 2013, precision and granularity figures remained high, though recall had dropped. For different kinds of obfuscation from 2012, recall remains low – though perhaps surprisingly is better for random obfuscation than for translation or summary.

Test	Plagdet Score	Recall	Precision	Granularity
02_no_obfuscation	**0.85884**	**0.83788**	**0.88088**	**1.0**
03_random_obfuscation	0.04191	0.02142	0.95968	1.0
04_translation_obfuscation	0.01224	0.00616	0.97273	1.0
05_summary_obfuscation	0.00218	0.00109	0.99591	1.0

3 The IPCRESS implementation

For the IPCRESS project, the previous codebase needed to be homogenized and developed in such a manner as to be scalable to very large datasets. The previous version/s were memory-based and thus not suitable for use at real scale (hundreds of gigabytes or more). The IPCRESS code has been fully re-designed as a disk-based approach, as an object oriented implemention in C++. A new stitching algorithm has also been developed.

The IPCRESS approach generates what we refer to as *secure stamps* from whole documents. From each stamp, we derive a set of shorter individual hash-like codes , I , from sets of words. These codes are considered irreversible. Individual codes are generated by using a sliding window of length, W_{len} , across the document stamp to extract that portion of the stamp in a manner similar to creating shingles. From this set of hash-like codes an index is populated with information related to the current window position within the document stamp and a document ID. This process is illustrated in figure 1.

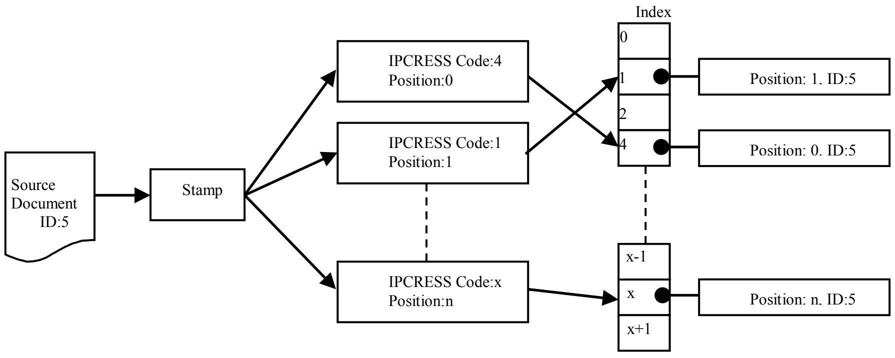

Figure 1. The indexing process using IPCRESS codes

A query for a suspicious document is similarly generated using the sliding window of length, W_{len}, over the stamp of the suspicious document to generate a set of IPCRESS code queries, Q.

Document ID and code position pairs, $i_i \in I$, are retrieved from the index, I , for each IPCRESS code, $q_i \in Q$, and sorted by document ID to give a set of results D. Each element, $d_{srcID} \in D$, relates to a source document and is itself a set of results, $T^{d_{srcID}}$; where, srcID, is a source document ID. Each set, $T^{d_{srcID}}$, is composed of information related to text segments, $t_j \in T^{d_{srcID}}$, each of length W_{len}; each element, t_j, is a pair composed of {suspicious position, source position}. This relationship is illustrated in figure 2.

Each set, $T^{d_{srcID}}$, is then reduced via a first stage stitching algorithm to produce a set of *run*s, $R^{d_{srcID}}$. Each run, $r_k \in R^{d_{srcID}}$, is 4-*tuple*, consisting of {suspicious start position, suspicious length, source start position, and source length}. This first stage stitching generates each run by finding consecutive elements of $T^{d_{srcID}}$ that are overlapping or consecutive in source position. Any runs, r_k, that are less than a defined minimum run length (MRL) are discarded.

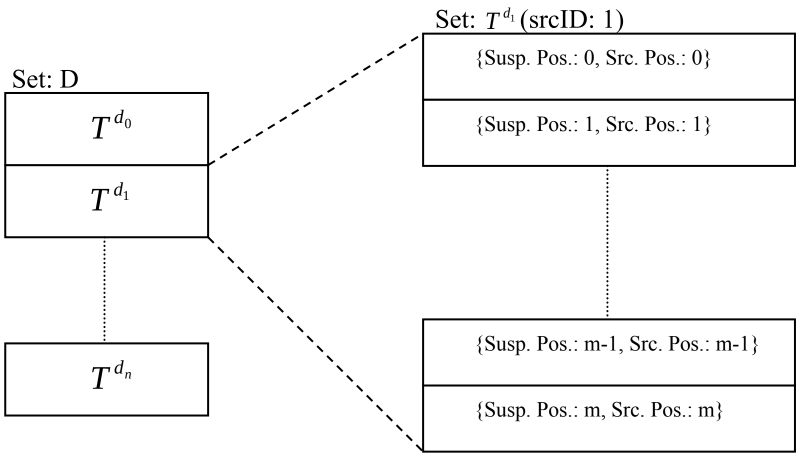
Set: T^{d_1} (srcID: 1)
Set: D
T^{d_0}
T^{d_1}
T^{d_n}
{Susp. Pos.: 0, Src. Pos.: 0}
{Susp. Pos.: 1, Src. Pos.: 1}
{Susp. Pos.: m-1, Src. Pos.: m-1}
{Susp. Pos.: m, Src. Pos.: m}

Figure 2. Ordering of results from document query process

A second stage stitching algorithm then produces a set of text *segments,* $S^{d_{srcID}}$, from the set of runs $R^{d_{srcID}}$. The algorithm finds subsets, $R \subset R^{d_{srcID}}$, such that each element of R are all within a defined *stitch distance* (SD) of at least one other element of R in terms of both suspicious *and* source position. The size of each subset is maximized so that $S^{d_{srcID}}$ is of minimal length. From each subset, R, a new *4*-tuple is formed, $s_m \in S^{d_{srcID}}$, that gives {suspicious start position, suspicious length, source start position source length}; the start positions are given by the first element of R and the lengths are determined from the last element of R. Any segments found that are shorter than a defined minimum segment length (MSL) are discarded.

3.1 Obfuscation Handling

We consider two initial obfuscation handling approaches based on transformations of a single query into closely related queries. The hash-like codes mentioned above are formulated such that code similarity can be indicative of data similarity, and as such the 'closeness' of any two queries can be based on binary distance approaches such as Hamming and Levenshtein.

The first approach, based on Hamming distances, generates transformed queries with a given maximum Hamming distance with relation to the original enquiry. For an original query $q_i \in Q$, of length W_{len}, this approach will generate an extra W_{len} transformations. For a Hamming distance of 1, say, this method involves an extra W_{len} look-ups for each initial query.

The second approach, based on the Levenschtein distance, similarly generates transformed queries from each original query, q_i. For a query length of W_{len}, this approach generates 2 sets of transformations T_i^0 and T_i^1 where i refers to a word position within query, q_i, and lies in the range $0 \le i < W_{len}$. This approach requires

2*W_{len} extra look-ups for each query. For each insertion the transformed query, T_i^n, is masked to length W_{len} for index compatibility.

4 IPCRESS vs previous PAN collections

Prior results offer up a standard to be achieved in re-implementation. The new codebase has been tested against data from PAN12 and PAN13, with modifications to the algorithm largely demonstrating slightly improved performance, as shown in the tables below:

IPCRESS raw – PAN12 data				
Test	**Plagdet Score**	**Recall**	**Precision**	**Granularity**
02_no_obfuscation	0.9437	0.9045	0.9877	1.0008
03_artificial_low	0.0956	0.0525	0.9942	1.0608
04_artificial_high	0.0200	0.0118	0.9852	1.2459
06_simulated_paraphrase	0.0992	0.0522	0.9922	1.0000
Obfuscation handler #1 (Hamming)				
02_no_obfuscation	0.9358	0.9048	0.9703	1.0008
03_artificial_low	0.1970	0.1110	0.9853	1.0178
04_artificial_high	0.0373	0.0201	0.9577	1.0759
06_simulated_paraphrase	0.1512	0.0825	0.9038	1.0000
Obfuscation handler #2 (Levenshtein)				
02_no_obfuscation	0.9236	0.9057	0.9423	1.0000
03_artificial_low	0.1888	0.1066	0.9820	1.0266
04_artificial_high	0.0682	0.0368	0.9489	1.0535
06_simulated_paraphrase	0.1345	0.0723	0.9572	1.0000

IPCRESS raw – PAN13 data				
Test	**Plagdet Score**	**Recall**	**Precision**	**Granularity**
02_no_obfuscation	**0.9253**	**0.9273**	**0.9233**	**1.0000**
03_random_obfuscation	0.1356	0.0729	**0.9675**	**1.0000**
04_translation_obfuscation	0.0243	0.0123	**0.9865**	**1.0000**
05_summary_obfuscation	0.0022	0.0011	**0.9959**	**1.0000**
Obfuscation handler #1 (Hamming)				
02_no_obfuscation	**0.9029**	**0.9289**	**0.8783**	**1.0000**
03_random_obfuscation	0.1297	0.1297	**0.9120**	**1.0000**
04_translation_obfuscation	0.0244	0.0244	**0.8953**	**1.0000**
05_summary_obfuscation	0.0035	0.0017	**0.9807**	**1.0000**
Obfuscation handler #2 (Levenshtein)				
02_no_obfuscation	**0.9058**	**0.9274**	**0.8853**	**1.0000**
03_random_obfuscation	0.2151	0.1224	**0.8936**	**1.0000**
04_translation_obfuscation	0.0743	0.0386	**0.9533**	**1.0000**
05_summary_obfuscation	0.0035	0.0017	**0.9920**	**1.0000**

5 IPCRESS vs PAN 2014 collections and Future Work

PAN 2014 test results showed expected granularity and precision, but a surprising difference between values for recall. Investigations led to the discovery of a bug in detecting UTF-8 codes; when applied to PAN 2012 and 2013 collections, a similar lowering of values was also observed. Further, our initial attempts at handling obfuscation show some promise, but much more rigorous evaluation will be required to determine the fullest extent of impact achievable by these approaches on the hash-like codes.

Test data	Plagdet	Precision	Recall	Granularity	Runtime
Corpus 2	0.28302	0.88630	0.16840	1.00000	00:00:55
Corpus 3	0.44076	0.85744	0.29661	1.00000	00:00:56

Through PAN 2014, we have demonstrated that the IPCRESS code produces results comparable to, and even slightly better than, the previous implementation, and effort has been put into ensuring the implementation is suited to scaling to very large datasets. These results are certainly not going to be anywhere near the best that is possible when evaluating similarity between texts where the content is fully exposed. However, that is not our challenge. and it is important to note again that our specific challenge is to be able to undertake plagiarism detection in such a way as would be impervious to a range of attempts to discover the content being matched against – a kind of privacy-preserving plagiarism detection that can be used against documents whose content should be kept from plain sight.

Acknowledgements

The authors gratefully recognize prior contributions of Neil Newbold, Neil Cooke, Peter Wrobel and Henry Cooke to the formulation of the codebase used for prior versions of this task, and by Cooke and Wrobel to the patents generated from these efforts. This work has been supported in part by the EPSRC and JISC (EP/I034408/1) and more substantially since PAN13 by the UK's Technology Strategy Board (TSB, 169201). The authors are also grateful for the efforts of the PAN organizers in system provision and managing the submissions.

References

1. Cooke, N. and Gillam, L.: Clowns, Crowds and Clouds: A Cross-Enterprise Approach to Detecting Information Leakage without Leaking Information. In: Mahmood, Z., Hill, R. (eds.) Cloud Computing for Enterprise Architectures, pp. 301-322. Springer, London (2011).

2. Cooke, N., Gillam, L., Wrobel, P., Cooke, H,, Al-Obaidli, F.: A high performance plagiarism detection system. In Proceedings CLEF 2011, Labs and Workshop, Notebook Papers, 3rd PAN workshop (2011).