Educated guesses and equality judgements: using search engines and pairwise match for external plagiarism detection

Notebook for PAN at CLEF 2012

Lee Gillam, Neil Newbold, Neil Cooke

Department of Computing, University of Surrey, UK
{l.gillam, n.newbold, n.cooke}@surrey.ac.uk

Abstract. This paper describes the approaches taken to the two subtasks of Candidate Document Retrieval and Detailed Comparison, in the Plagiarism Detection track at PAN 12. For the first of these, we describe how we used a combination of frequency and a variation of a contrastive corpus measure to select keywords with which to make queries to the ChatNoir search system; for the second, we provide an overview of how we re-used software that had previously featured in PAN 11. We comment specifically on how effective both approaches were, and what steps we might take to improve if the competition remains substantially similar next time.

1 Introduction

The PAN activity first appeared in 2007 as an International Workshop on Plagiarism Analysis, Authorship Identification, and Near-Duplicate Detection, and subsequently evolved its name to Uncovering Plagiarism, Authorship, and Social Software Misuse[1]. The first competitive activity in PAN occurred in 2009, separated into an external task that involved checking document content against a collection, and an intrinsic component apparently looking at writing style changes within a document and which seems to have migrated into the authorship task. External detection is consistent with what many would typically think of as plagiarism detection – finding sources that match parts of the content of a particular document - and it is this task with which we are largely concerned in this paper.

The external detection part of the plagiarism task remained relatively consistent from 2009 to 2011, treating a collection of (a few) tens of thousands of documents in almost equal quantities of suspicious documents that may contain plagiarized material and source documents from whence the this material may have been taken. It was useful first to construct some kind of index of the set of source documents, and then to use this to respond to queries generated from the suspicious documents. For example, the n-gram based approach of Grozea and Popescu 2011 would seem to suggest an

[1] Presumably the N of PAN now comes from the conjunction.

inverted index keyed on n-gram[2], with initial result ranking according to the number of matches in each source document and a threshold above which further analysis is undertaken. Clearly, for such n-gram based approaches, the size of the index will depend on the value for N and the extent of overlapping; the speed of match will depend on how many n-grams are selected from the suspicious document. With the number of overlapping n-grams that could be created from just one document, and subsequent analytical steps, it is easy to understand why various researchers would want to make use of high performance clusters to undertake such tasks.

In contrast to these previous iterations of PAN, in 2012 the plagiarism detection tasks seem to be encouraging a search-engine-first keyword-based approach, with subsequent checking. Here, the subsequent checking could be undertaken interactively, in reducing the quantity being checked once a “hit” is obtained, or by constructing a sub-index of all the retrieved material or, as seems to be implied, by pairwise checking (Elsayed, Lin and Oard, 2008). Part of the rationale for this shift seems to be the difficulty that previous participants have had in processing the relatively small (GB) collections of data, which would make scaling to larger (TB) collections quite onerous. There is already a multitude of online resources that claim to detect plagiarism and are built above common search engines. But such systems seem to operate best when extended phrases are used. Of course, extended phrases come at additional cost to the search engine provider creating a tension between accuracy and utility. This also presents a very different problem: where previously an exhaustive match could be made within the entire collection, exhaustive match now depends on the ability to make the search engine return a set of results from its index that are useful for this purpose. The simplest way to cater for an exhaustive match is to make more queries until the gain achieved is suitably diminished. The costs of undertaking such a task are therefore devolved into the costs of search, and each query-response-retrieval will take some time, and the cost of match with this resulting subset (and subsequent cycles of these two as required). But if the former fails (recall=0), the latter is not possible.

In this paper, we outline the approach taken at the University of Surrey to these two tasks of Candidate Document Retrieval and Detailed Comparison in the Plagiarism Detection track at PAN 12. In section 2, we describe how we use a combination of frequency and a contrastive corpus measure to select keywords with which to make queries to the ChatNoir search system and discuss the results obtained, which show high values for recall offering good scope for the match phase. In section 3, we provide an overview of how we re-used software that had previously featured in PAN 11 and comment on several simple optimizations that would have reduced quite significantly the time taken for our comparisons. We decided against optimization as this would have moved us away from our goal of developing a scalable indexing system. Section 4 concludes the paper with considerations for future work, including how we envisage processing the entire ClueWeb collection for full-document search and the initial steps we have made towards this.

[2] However, their paper does not readily mention the value of N used for their competitive effort (the associated presentation suggests N=2,3 .. 16). Without such information it’s not readily possible to estimate the size of their index or challenge in building it.

2 Candidate Document Retrieval

Candidate Document Retrieval involves creating a set of queries for a text that might be useful in retrieving other texts from a search engine that offer matches to that text. The extent to which an individual text retrieved in such offers a match to the original can only be known through subsequent processing. In this case, the search engine is Chatnoir, developed by the Webis Group at Bauhaus-Universität Weimar, which indexes the ClueWeb09 collected in January and February 2009 and comprising of some 1,040,809,705 web pages in 10 languages (25 TB of uncompressed data)[3].

In formulating our approach, we explored the extent to which several extant text analysis components of the System Quirk toolset could offer something for such a task. We considered how to make use of n-grams (here we might include term-bearing, but also collocation patterns and concordances), frequency analysis, contrastive corpus analysis, and indicative text summarization, all of which are variously offered through the applications Ferret, ColloQator, KonTEXT and Summ-it.

Our initial efforts suggested little gain from indicative summarization, although this will be worth exploring again now that our approach produces a reasonable return. And since we were unable to identify how phrases could be used with Chatnoir, this seemed to enforce an approach based entirely on locating keywords. Furthermore, initial tests with Chatnoir showed some unexpected outcomes. Consider, for example, the text with ClueWeb ID 255104308; this is the first response to a query comprising the two words *flushmate* and *gpf*, and contains four instances of the first word and two of the second in about 500 words. The whole text is part of a product catalogue with numerous outbound links and just one contiguous paragraph of text that contains neither of these terms. The second result contains 36 instances (30 and 6 respectively) in about 600 words. Moreover, the first term appears sooner in the second document than in the first. The ratios and positions seem unusual, suggesting either that word count might be being produced after removing data such as prices (formed of numbers and punctuation), or there is an unclear interplay between the ranking function (BM25) and the term proximity approach, about which it is not possible to find details of how bucket sizes are produced for Chatnoir. Having observed this, and given the likely passage-based formulation of the document set, it was considered that extracting terms at document-level would be doubly unlikely to obtain good results, and so subsequent efforts would work on smaller fragments to see if proximity could be exploited.

Core to our approach is enhanced weirdness (*ew*, eqn.1), obtained by squaring the relative frequency in our scaled weirdness equation (e.g. Gillam, Tariq and Ahmad, 2005). Scaled weirdness has been used variously as a contrast between relative frequencies in general and specialist language to flag terms; here its purpose is to generate sets of search terms which have a lower likelihood of appearing in general text and therefore would be expected to occur in fewer documents in an index.

[3] See: http://lemurproject.org/clueweb09.php/ [accessed, 14/8/2012]

$$ew = \frac{N_{GL} f_{SL}^{\ 2}}{(1 + f_{GL}) N_{SL}^{\ 2}} \quad (1)$$

where f_{SL} is the frequency of a word in the (split) text, f_{GL} is its frequency in the 100m tokens of the British National Corpus (BNC), and N_{SL} and N_{GL} are the token counts of the (split) text and the BNC respectively. This is used in the approach briefly outlined below:

```
For each suspicious text, T:
  1. Split to sub-texts S by number of lines l.
  2. For each sub-text in S, generate queries Q by:
       a. Rank by ew.
       b. Select the top 10 terms, and re-rank by frequency
       c. top frequency-ranked word paired with the next m words
  3. Retrieve texts for each query in Q.
```

Consider the first text in the test collection (004 – Table 1) without line splitting applied, and ranked by ***ew*** to obtain the top 10 terms (2a, above) . The table demonstrates how this promotes certain terms that are both highly frequent and unusual (*toilet* and *toilets*), which by weirdness values alone would not feature in this table; a frequency of 4 is sufficient for *caulk* to feature, showing the bias towards weirdness. Re-ranking list by frequency would select *toilet* for pairing with the others, and the first query of *toilet* and *toilets* (2c) to use to retrieve texts (3).

Term	f_{SL}	f_{SL}/N_{SL}	**w**	**Ew**
gpf	9	0.001403	140401.2	196.9161
flushmate	5	0.000779	78000.65	60.77657
toilet	***161***	***0.02509***	***1590.64***	***39.90853***
caulk	4	0.000623	62400.52	38.897
actuator	8	0.001247	31200.26	38.897
flange	20	0.003117	10064.6	31.36855
shims	6	0.000935	31200.26	29.17275
toilets	***64***	***0.009974***	***2021.069***	***20.15715***
inducer	8	0.001247	13866.78	17.28756
composting	13	0.002026	7511.173	15.21665

Table 1: Enhanced weirdness applied to suspicious document 004 in the test collection

Our competition run used ***l***=25 (lines), ***m***=4. Creating the queries takes just 4 minutes for the entire set of 32 test texts. Processing is readily automated via a set of (Linux) shell scripts that make use of split and KonTEXT, and formulate the queries to Chatnoir, obtain responses in JSON, process the JSON to obtain the required LongID, retrieve the texts, and pass them to the pairwise matching. For this last part, we use our own pairwise comparison approach, outlined in the next section, to select resources to submit for evaluation.

For suspicious-text-004 from the test collection, we craft 90 queries which retrieve some 729 files. 21 of these files contain matches of various sizes, and with a degree of duplication (see Table 2). We report all matches.

ID	*su_offset*	*su_length*	*so_offset*	*so_length*	*Notes on duplicates*
82916556	315	5665	1149	4923	Contained: 340124840 (622+1326); 82916586 (1868+337)
	12584	1412	7725	1367	
	36534	330	6583	340	
82916557	3711	8204	10	6193	Significant overlap: 655702818, 926517445, 1037219213, 1082516754, 1234439206, 1283114388 (all 4924+3705); 512814224, 1337033767 (4933+3641) Contained: 456806343 (4933+3135)
32718446	17202	1040	632	1499	
811900	19134	7285	3034	7233	Exact: 811901, 811902
102839362	30070	6472	10	8732	Contained: 476400740, 601432982 (30301+4339)
74735759	37035	431	5752	438	

Table 2: Detections applied to suspicious document 004 in the test collection. Results are reported by start position in each text (offset) and length of detection (length), with prefixes indicating whether this was in the suspicious (su_) **or source document** (so_).

The suspicious file is 37472 characters. Of note in these results:

1. Overlap and duplication can be significant
2. Large overlap between first segment of 82916556 and the result for 82916557.
3. Large undetected segment from 11915 to 17202

Using this approach, we achieved the highest values for recall (0.5567) of downloaded and retrieved sources amongst the competitors when including near-duplicates(see Table 3), though a near-duplicate is as yet undefined.

Reported Sources		**Downloaded Sources**		**Retrieved Sources**	
Precision	Recall	Precision	Recall	Precision	Recall
0.6266	0.2493	0.0182	**0.5567**	0.0182	**0.5567**

Table 3: Precision and recall values for our approach; 0.5567 was the highest recall value achieved in the task..

However, these results are not necessarily a reliable indication of performance – a second (unreported) attempt was made to see whether a variation to ***l*** and ***m*** might improve performance; quantities of downloads likely reflects the extra work done here, and is also likely to be a factor in what otherwise appears to be an under-reporting of sources that constrained our precision at 0.2493 (against a possible 0.2775).

3 Detailed Comparison

In Cooke et al (2011) we described various aspects of our system as used for the external plagiarism detection task, which we stated could process the entire PAN11 collection within relatively short timescales, and which was still able to produce a reasonable degree of matching performance (4th place, with PlagDet=0.2467329, Recall=0.1500480, Precision=0.7106536, Granularity=1.0058894). We also stated that we were unable to disclose too many details about the approach due to a patent application that was in progress. The patent was since filed in the US (US13/307,428, filed 30th November 2011), but we are waiting for the review of that filing before disclosing the simple method used at its core.

What we can state at this time is that we do not:

1. remove stopwords *per se* since we consider them to be an important part of the signature of the text (our approach to one part of the authorship attribution task builds out our consideration of this importance, albeit in a rather different way).
2. use methods of encryption or hashing in order to create same-length keys for the data
3. break the text into large numbers of short character-based or word-based n-grams.

Indeed, we consider that such approaches have a relatively high computational cost which rapidly become prohibitive when dealing with large volumes of data (e.g. if we were attempting to deal directly with the ClueWeb09 data).

Our approach uses the same parameters and values as for PAN11. Parameters that we could tune were:

- Minimum detection run length (RR) – to remove segments less than 50 words
- Maximum Stitch distance (SS) – to address granularity in joining segments
- Minimum cosine score (CS) – to verify segment similarity.

We kept the value of these parameters consistent with those used in PAN11 for comparison purposes, and since our previous parameter sweeps had not demonstrated much by way of gain across a range of values. The values used were: RR=50 (minimum suggested length of plagiarism); SS=900, CS=0.75.

Our approach to translated texts merely made use of a post run adjustment by character ratio of the source to the translation via a shell script run subsequent to matching to modify the character positions in the XML results.

The software was constructed in a relatively ad hoc manner previously, using a combination of shell scripts, Python and C++ code. We only put effort into forming this into batch programs, which leaves a large number of inherent overheads in the interfacing of components – e.g. launching a shell to launch Python code that in turn loads in a shared object file and coverts calls from Python to C++ for its operation, and then runs other separate components, for example, for our stitching approach and cosine matching – each of which involves another intermediary file-based communication. Added to this processing cost, our first pass search usually derives matches from large collections and builds up an index from this and the cosine matching reopens those files implicated in order to undertake verification. So where this is a match, the files are being processed twice. Comparing pairs in the training corpus took, on average, 7.8 seconds but are reported at 9.4 seconds for the test corpus on an apparently more capable system. Since many plagiarism detection systems in previous years had reported relatively slow processing of large collections, we did not perceive a need to optimize our code for speed although clearly there is plenty of scope for this and we would expect at least half the processing time to be necessary. We also make no attempt to use threads or multiple cores to achieve better throughput.

The software was provided, under licence, to the organizers for evaluation purposes. The Zip file containing the program occupies around 240kb, and requires python 2.7.1. It was built for a 64-bit Ubuntu platform and appears to have been usable by the organizers without requiring modifications to the build.

Performance results from the training corpus are shown below (Table 4). We did not produce results for '05_translation', as this was being handled differently in the test phase.

Test	**Plagdet Score**	**Recall**	**Precision**	**Granularity**
01_no_plagiarism	1.0	1.0	1.0	1.0
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

Table 4: Performance results for the training corpus. Note that we have yet to fully address the problem of obfuscation, hence low values in recall.

Our competition results were largely as expected. We achieved the highest precision, and 5th best granularity, but low recall (Table 5).

Detailed Comparison Task

Rank	**PlagDet**	**Precision**	**Recall**	**Granularity**	**Runtime* [Seconds/Pair]**
9	0.3088109	**0.8984268**	0.1903951	1.0243572	9.4009198

Table 5: Performance results for the training corpus. Note that we have yet to fully address the problem of obfuscation, hence low values in recall.

4 Conclusions and Future Work

In contrast to these previous iterations of PAN, the 2012 external plagiarism detection tasks seem to be encouraging a search-engine-first keyword-based approach, with subsequent checking. The ability to undertake match, then, depends on the educated guesses made of suitable queries that will impel the search engine to offer up the right documents. It is not possible to recover from bad guesses, only to keep guessing in the hope that something will be found. Whilst the approach to crafting the guesses can be made systematic, obtaining results depends on the extent of pollution/noise contained by the search engine – i.e. the number of results that would be produced ahead of the results sought in each case. It is quite possible, also, that constraints within the search system or the implementation itself would prevent a specific text being returned for a particular query. In addition, it could be quite possible to produce good pairwise match results in relatively short time without really performing pairwise match – e.g. using a bag of words or n-gram approach when sentences are within a few words length of each other, but as previous PANs have shown, not being able to readily scale such an approach.

Our results have shown that we have a decent strategy for educated guesses, but that our pairwise matching suffers under obfuscation. We could readily reduce the number of queries required by dropping the need to query for segments already covered by results; on the other hand, we should look to formulate more queries for an unmatched segment. For Pairwise matching, and for our approach in general, we need to begin handling obfuscation. However, such approaches are not really in our preferred direction of travel, which is towards full-document (private) search. And having obtained ClueWeb09 dataset, and formed an approach for this which we believe will readily scale, hope to be able to report on this at the next PAN.

Acknowledgements

The authors gratefully acknowledge the prior contributions of Peter Wrobel and Henry Cooke to the formulation of the codebase used for this task. This work has been supported in part by the EPSRC and JISC (EP/I034408/1), and we are very grateful to Amazon Web Services (AWS) for providing a supporting grant for this research and for competition use of both EC2 and EBS services.

We also gratefully acknowledge the invaluable efforts of the PAN12 organizers in crafting and managing the tasks.

References

Cooke, N., Gillam, L., Wrobel, P., Cooke, H. and Al-Obaidli, F., 2011, A high performance plagiarism detection system. Proc. of the 3rd PAN workshop.

Elsayed, T., Lin, J. & Oard, D.W., 2008, Pairwise document similarity in large collections with MapReduce. Proc. 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers, pp265-268

Gillam, L., Tariq, M. and Ahmad, K., 2005, Terminology and the Construction of Ontology. Terminology 11(1), pp55-81. John Benjamins Publishing Company. ISSN 0929-9971; E-ISSN 1569-9994