A high-performance plagiarism detection system

Notebook for PAN at CLEF 2011

Neil Cooke, Lee Gillam, Peter Wrobel, Henry Cooke, Fahad Al-Obaidli

University of Surrey

N.Cooke@surrey.ac.uk, L.Gillam@surrey.ac.uk, Peter.Wrobel@gmail.com, Henry.Cooke@students.plymouth.ac.uk, fa00057@surrey.ac.uk

Abstract. In this paper we report on our high-performance plagiarism detection system which is able to process the PAN plagiarism corpus for the *external* plagiarism detection task within relatively short timescales in contrast to previously reported state-of-the-art, and still produce a reasonable degree of performance (PAN 11, 4th place, PlagDet=0.2467329, Recall=0.1500480, Precision=0.7106536, Granularity=1.0058894). At the core of our system is a simple method which avoids the use of hash-type approaches, but about which we are unable to disclose too many details due to a patent application in progress. We optimised our performance using the PAN10 collection, and used the best parameters for the final submission. We anticipated a relatively similar performance at PAN11, modulo changes to the plagiarism cases, and 4th place this year put us between participants who had been 5th and 6th in PAN 10.

1 Introduction

According to a Detica and Cabinet Office report on the UK, the annual cost of theft of intellectual property is £9.2 billion, and "this cyber criminal activity is greatly assisted by an 'insider'"[1] Whilst organisations may have rules and policies geared towards prevention, without computational prevention the emphasis remains on detection. Here, difficulty arises in that companies need to be able to search both the web and the document repositories of collaborators, partners, competitors etc. without revealing the content of their query, and in a highly efficient way. In principal, a proxy could be used to hold corporate repositories of different companies, and searched to identify shared content that might be of concern to collaborating organisations. However such a proxy is unlikely to be trusted by all without considerable and provable security constraints on the proxy. A means that allows each to hold their own, provide indications of shared content, and yet not reveal the query, is required. Further, an indication of shared content might suggest an IP infringement

[1] http://www.cabinetoffice.gov.uk/sites/default/files/resources/the-cost-of-cyber-crime-full-report.pdf

of which investigation would be necessary yet the indication must not reveal the IP involved, and it needs to be up to the organisations concerned to determine how to resolve this.

The reason we have been investigating the applicability of plagiarism detection approaches to such a problem should be apparent. However, typical plagiarism detection approaches prove unsuitable for a number of reasons including (i) computational inefficiency in producing useful patterns; (ii) brittleness to single character variations in strings; (iii) key proximity not offering a useful indication of data similarity; (iv) susceptibility to brute force attacks. Some use small-stepped overlaps (shingles) to attempt to address brittleness; this needs additional hashing operations which increase computational inefficiency, increase susceptibility to brute force attack, and does nothing for key proximity. We have been attempting to develop an approach that overcomes all four of these problems, and our research to date suggests we are being successful. When we set aside systemic overheads (computer setup, translation, XML production, remapping for translation, zip and send), it takes a mere **12 minutes** to process the *entire* PAN 11 collection using a moderately specified server, we have addressed certain kinds of brittleness, our key proximity is meaningful, and brute force attacks are rather more difficult. However, due to a patent application it is not presently possible to disclose how the approach works or to offer code which would demonstrate. We intend to inform interested parties related to the PAN competitions of how this works at a later time – and have some confidence that the simplicity of the approach will be refreshing. Our approach draws inspiration from early work in information retrieval [1].

2 External Plagiarism Detection

Our system has been built in Amazon Web Services (AWS), so we can deploy our system as required, have dedicated use of the required resources for that period, and can release them when they are no longer required with costs of participating in PAN quantifiable in relation to our resource use. This offers clear incentives for a computationally efficient approach, though we have been fortunate in obtaining a grant from Amazon for use of AWS so we also did not need to pay for our use.

We used ***one*** virtual core in a ***single*** High-Memory Quadruple Extra Large Instance (m2.4xlarge) instance. An Amazon m2.4xlarge costs $2 per hour, and offers the following specification[2]:68.4 GB of memory; 26 EC2 Compute Units (8 virtual cores with 3.25 EC2 Compute Units each); 1690 GB of instance storage; 64-bit platform. A key benefit of using AWS was that we could rapidly scale the computing resource to the problem space and identify the right size of server needed for competition purposes via trial and error. Programs, data, and results, were placed into a 100GB machine-independent Elastic Block Storage (EBS) to allow for this. Had we only needed to use AWS for the time from release of data until our first results were submitted, without treatment of translations, we would certainly have spent no more than $4. Even with overheads, our subsequent runs are relatively inexpensive.

[2] http://aws.amazon.com/ec2/instance-types/

During each PAN 11 run, the memory load increases to about 17GB, which is very safely below that available. Furthermore, the process runs on single core – whilst parts of the work could be parallelized across the cores, we did not see an immediate performance gain for implementing this in contrast to efforts needed; nor did we see a need to construct a cluster for our purposes. Our system has been composed from a variety of shell scripts, Python, Java and C++, and external libraries and APIs, and is configured and run as follows:

1. Create and start an Amazon EC2 instance that uses a Canonical Ubuntu image, has ports 80 and 22 open to allow for http and ssh traffic from required hosts as necessary, has associated keypair information, and access to the working team
2. Use Ubuntu's Advanced Packaging Tool (APT) to install required external software libraries
3. Add a 100GB EBS volume to this instance and unpack the competition data.
4. Download, install, and use language identification software such as TextCat against the complete PAN 11 dataset
5. Use a machine translation engine such as Google Translate to convert text to English, replacing translated texts into the source files and noting size differences from originals by number of characters.
6. Index the source files.
7. Run the detector to discover instances of plagiarism against the index.
8. Run the stitcher to reduce granularity. This is a blind stitching approach, i.e. only uses metadata - filename, run length and position with run lengths greater than a given size, within a maximum spanning distance, stitched into one run.
9. Produce competition XML applying cosine distance measures on the fly to validate passages.
10. Modify the XML to adjust offset and length information for translated texts.
11. Submit results

Steps 1 to 3 to set up the system can be readily undertaken in under 15 minutes. This need only be undertaken once.

Steps 4 and 5 ensure comparability with others using such translation. Language identification requires each text to be processed, and runs at about 90 texts/minute. On a single CPU, this means about 5 hours (26,000 texts ~ 289 minutes) for language identification and results in some 890 texts identified for translation. Translating 890 identified texts took 5 hours 15 minutes via Google Translate. This need only be undertaken once.

Steps 6 to 8 represent the core of the method for detection purposes, and require **12 minutes** to process the entire collection. This speed means various values of the different parameters can be tested quickly by iterating these steps. Stitch time per PAN11 run is approximately 30 seconds.

Step 9 adds about 45 minutes per run in conversion, computation and filtering to create the valid XML for each run. Cosine calculations are used on short runs – long detections typically imply high confidence – and tap the data during the conversion, so add minimal processing time. Cosine distance use also bears novelty: we use a

fixed set of frequent words and prepare counts during mapping to save file overheads and avoid the cost of producing detection-specific cosines.

Step 10 is complementary to steps 4 and 5 for transforming character positions and counts in the ratio indicated by respective file sizes. This adds 5 minutes per run.

Step 11 adds about another 5 minutes to the entire process per run, depending on the speed of the submitting user.

The most time-consuming work is being done in relation to translation, requiring almost 10½ hours to complete relevant processes – over 10 times the duration of all other required efforts combined.

In contrast to **PAN 10 competitors**: our indexing, detection and format conversion takes half the time of the PAN10 competition winners and uses just 1 core instead of their 24 [3]. Our resource requirements are also lower than the threaded use of 32 cores by last year's runners up [3]; we cannot comment on run times of the 3rd placed competitors as runtime is discussed but not made explicit [5]; and we are more efficient than the 40 hours taken using 8 cores by the 4th placed competitors [6].

Our competition entry used detection runs over 50 words, and a cosine distance of 0.757. One of our pre-final submissions could have achieved a marginally better overall score (0.2485299) but this would not have changed the final rankings.

3 Evaluation

We initially tested our approach internally using just 100 of the PAN09 suspicious files with their associated reference material (i.e. not the whole PAN09 sources). On completion of this phase, we transferred our system to AWS and carried out inspectable experiments on PAN10 data using either the first 1000 suspect files, or the 100 largest suspect files, before undertaking full runs.

Parameters that we could tune were:

- Minimum detection run length (RR)
- Max Stitch distance (SS)
- Min post-stitch run length (PR)
- Min cosine score (CS)

Tests with PAN10 data suggested initial values for our parameters of RR=50, SS=900, PR=50, CS = 0.757, or PR 55 with CS = 0. Following translation of PAN10, we ran parameter sweeps around these values to determine best performance. This suggested: RR=50 (minimum suggested length of plagiarism); SS=900, PR=50, *CS=0.75*. The combination offered a PAN10 score of **0.658429611476**, with many other scores above 0.65 suggesting there was little more that could be optimized at this point. Notionally, this overall score would have achieved 4th place in PAN10 (3rd: 0.6984; 4th, 0.6209).

These parameter values were used for our final PAN11 run, giving us 4th place in external detection with:

PlagDet	Recall	Precision	Granularity
0.2467329	0.1500480	0.7106536	1.0058894

Against PAN11 annotations, one of our pre-final submissions could have achieved a marginally better overall score (**0.2485299**) using a different approach to stitching. This had been a speculative approach and had not produced compelling results against PAN10; it also would not have changed the final rankings.

4 Conclusion

In our first foray into this competition, we are reasonably satisfied with achieving 4th place in external detection. We inserted ourselves between the 5th and 6th placed competitors from PAN 10, suggesting that we have a reasonably competitive approach that also suggests computational efficiency, lowered brittleness, key proximity that indicates data similarity, and robustness to brute force attacks. Our approach is able to undertake the core activity of plagiarism detection at high speeds - 12 minutes for the entire PAN11 dataset and similar efficiency on previous PAN data, whilst some reportedly find it difficult to cope with processing this volume of data. We would like to think that if we can deal with a few gigabytes within such a time on one core of a large server, we could readily scale our approach to much bigger data collections and potentially the entire web. In another experiment, reported in [2], we investigated news reuse in texts from Reuters Corpus Volume 1 (RCV1 [7]) which comprises a year's worth of English news output in 1996/7. We compared 750,000 x 750,000 files, using an m2.xlarge EC2 instance, and reported all cross-copying between news reports. Processing took approximately 36 minutes, and there were a reasonable number of indications of content reuse by Reuters' journalists.

There was a tension between being able to process the data quickly, and performing well within finite time. Without translation, we were ready to submit our initial results within two hours of the data release, which would have cost a mere $4 in compute time. It would have been possible to complicate our approach with more demanding computations, but processing time and cost would be a critical factor for an internet-scaled application, and we prioritized achieving good results quickly rather than, say, waiting for vastly extended periods only to achieve slightly better results.

There are many possibilities for improving our system's detection performance. A more efficient translation alignment approach would be one, but such alignment will slow our system per run. There are also different search strategies we might deploy in order to deal better with some of the obfuscation approaches, but again this will come at a cost. We can clearly improve our XML production processes and filtering approach. Whether the benefits of these improvements outweigh the costs of development and speed is something that we cannot yet answer. Clearly there is plenty of room for improvement to our detections, and deeper investigation into the characteristics of the PAN11 dataset will help here. One thing we are convinced of is that we know ways in which to improve, and we may have several possibilities for the exploitation of such a system.

5 Acknowledgements

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 sterling efforts of the PAN11 organizers in managing the competitions, and the helpful contributions of Surrey students David Cheung and Ankush Sharma to our understanding of stitching approaches.

6 References

[1] Luhn, H. P.: A statistical approach to mechanized encoding and searching of literary information. IBM Journal of Research and Development, 1(4), pp309-317, 1957.

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

[3] Kasprzak, J. and Brandejs, M.:Improving the Reliability of the Plagiarism Detection System - Lab Report for PAN at CLEF 2010.

[4] Zou, D., Long, W-j. and Ling, Z.: A Cluster-Based Plagiarism Detection Method - Lab Report for PAN at CLEF 2010.

[5] Muhr, M., Kern, R., Zechner, M. and Granitzer,M.: External and Intrinsic Plagiarism Detection Using a Cross-Lingual Retrieval and Segmentation System - Lab Report for PAN at CLEF 2010.

[6] Grozea, C. and Popescu, M.: Encoplot-Performance in the Second International Plagiarism Detection Challenge - Lab Report for PAN at CLEF 2010.

[7] Rose T., Stevenson M., and Whitehead M.: The Reuters Corpus Volume 1 – from Yesterday's News to Tomorrow's Language Resources. In Proceedings of the Third International Conference on Language Resources and Evaluation, 2002.