External Plagiarism Detection Based on Standard IR Technology and Fast Recognition of Common Subsequences

Lab Report for PAN at CLEF 2010

Thomas Gottron

WeST – Institute for Web Science and Technologies
University of Koblenz-Landau
56070 Koblenz, Germany
gottron@uni-koblenz.de

Abstract The plagiarism detection system described in this paper is aiming at bringing external plagiarism detection to the desktop. The main ideas are to incorporate standard IR technologies for the candidate selection and efficient data structures for the detailed analysis between a suspicious and a candidate document. Given that the system so far has only reached prototype status, the first results look promising.

1 Introduction

The plagiarism detection system described in this paper was designed following two main aims. First, to implement candidate selection based on standard IR engines. The intention of this aim is to substitute the candidate selection used for the PAN competition with the search API of a web search engine. In this way it is easily possible to lift the competition system to a real world scenario, e.g. to check essays or term papers at university courses for plagiarism from the web. The second aim was to design a system that can run on commodity hardware. Also this intention was motivated by practise, where teachers in educational institutions do not have access to high-end computing machinery, but rather run plagiarism detection software on their laptop or desktop machines.

Both aims were realised. The candidate selection process was implemented based on using Lucene with an out of the box configuration. By using an efficient data structure to mine similar sub-sequences from a pair of documents, also the detailed analysis is running remarkably fast. Scaling the run-time to a scenario of analysing 20 student papers of 10,000 words each, the whole process can be completed in less than one minute, which is acceptable in practise.

1.1 Related Work

In last years competition, Zechner et.al. [15] was the only team employing a standard model of textual IR for candidate selection. The source documents were indexed on a

sentence level and sentences of suspicous documents were used as queries. Similarity was calculated via the well established cosine measure. The mapping between sentences from the suspicious document to the sentences of source documents also provided the alignment of similar subsequences.

All other approaches used task specific index structures. Malcolm and Lane [9] used the desktop plagiarism detection system Ferret, which is based on common word tri-grams. Word n-grams are also the method of choice of the WCopyFind tool employed by Vallés Balaguer [13]. Instead, Basile et.al. [1] encoded texts as a word length sequence and used a downstream vector-based n-gram distance measure for candidate selection. Kasprzak et.al. [6] incorporate common text shingles in the pre-selection process and Shcherbinin and Butakov [10] employed hash-based fingerprints for candidate retrieval. A very different approach was taken by Grozea et.al [3]. They used string kernels to compute a complete similarity matrix for each pair of source and suspicious document.

Finding similar subsequences in strings is a problem commonly addressed in bioinformatics. Algorithms finding longest common subsequences, like the Hirshberg Algorithm [4] are based on the Levenshtein edit distance [7] and aim at finding a globally optimal alignment. The Smith-Waterman algorithm [11] is capable of finding also locally similar subsequences.

In the context of plagiarism detection, Kasprzak et.al. [6] detect regions that are densely populated with common shingles in both documents. Malcolm and Lane [9] compute the Jaccard coefficient for similarity on word sequences. A detailed analysis based on a T9-like word encoding and detecting and merging squared shapes in the dotplot [8] representation was employed by Basile et.al. [1]. Grozea et.al [3] have developed an alternative to dotplots: the encoplot representation. The detailed analysis is based on detecting continuous lines in the encoplot.

Concerning performance optimization, the most common approach is to parallelize the algorithms. Grozea et.al [3] followed this road and Kasprzak et.al. explained in [5] in detail how their pre-selection mechanism has been extended to a distributed version running on a cluster-architecture.

2 External Plagiarism Detection

The plagiarism detection system presented in this paper follows the standard designed as described e.g. in [12]. When provided with a suspicious document the pre-selection component uses the Lucene engine to retrieve candidate documents from the source collection for a detailed comparison. The detailed analysis then provides tuples of sequences from suspicious and candidate documents that already represent detected plagiarised contents. A series of post-processing filters takes care to remove pathological cases.

Prior to building the Lucene index, all non-English documents were translated into English using Google's translation-service. Essentially, this corresponds to a standard cross-language indexing approach. To be able to easily map the translated parts back onto the original texts, they were translated in small chunks of a few paragraphs. The

information which parts of the texts correspond to each other was stored for a later on backward resolution of character positions in plagiarised parts.

2.1 Pre-Selection

The Lucene index for pre-selection covers only English documents – either English originals or the translations of documents written in other languages. To cope with Lucene's feature of indexing only the first l token in a text, each document d_n was split up into smaller parts d_n^m of a fixed length of $l_d < l$ terms. Each part d_n^m was submitted to the indexing engine as an individual document, but with a reference to the source it was taken from.

A similar step was taken for finding candidate documents to compare with a given suspicious document. The suspicious document s_i was split into even smaller parts s_i^j of length $l_q < l_d$. These parts were used as queries to retrieve relevant documents from the index. For each query the top-k results are collected, that have scored a relevance value $\rho_{\text{Lucene}}(s_i^j, d_n^m)$ above a certain threshold value θ. This process is iterated over all parts s_i^j of a suspicious document. For each source document d_n a total candidate score $c(s_i, d_n)$ is computed by summing over all obtained scores for all parts of that document and all parts of the suspicious document, so:

$$c(s_i, d_n) = \sum_j \sum_m \rho_{\text{Lucene}}(s_i^j, d_n^m) \tag{1}$$

Ordering the source documents by decreasing candidate scores $c(s_i, d_n)$ creates a ranking of candidate documents for a particular suspicious document s_i. This ranking is the input to the next step: the detailed analysis.

2.2 Detailed Analysis

The ranking computed by the last step has the advantage of assigning a priority to the candidate documents. This can be used to balance time constraints of a fast analysis against completeness, simply be considering more or less documents of the ranked list in a detailed analysis.

The comparison of a pair consisting of a suspicious and a candidate document comes down to finding similar local token sub-sequences. Talking about tokens in this case is intentionally general, as it can be realised in different ways. The methods described here can be applied to tokens that correspond to characters, terms or larger constructs like n-grams. The setup described in this paper eventually used sorted term 5-grams, where the purpose of sorting the terms in each 5-gram was to partially overcome local changes in word order.

The problem of finding similar local sub-sequences is commonly known in bioinformatics, where DNA sequences have to be compared for similar fragments. Hence, the first approach was to apply the Smith-Waterman algorithm [11]. But, given its quadratic complexity, it can only be run on small documents or, again, parts of documents. This turned out not be suitable for a plagiarism detection scenario with book-length documents. We investigated applying Smith-Waterman to local areas in the token

sequence around cooccurrances of word n-grams. This approach lead to low values for both, recall and precision, and high values of granularity (see the results in section 3). Also, execution time was still around twice as long as for the solution we eventually applied.

Another typical approach is to plot the positions at which each token from a candidate document occurs in a suspicious document. This leads to a graph like the one in Figure 1, sometimes referred to as a *dotplot* [8]. While random cooccurrences appear as noise in this kind of plot, regions of plagiarism point out as longer lines. Such a plot can be generated efficiently, by building an inverting index of tokens and their positions in the suspicious document and then looking up each token from the candidate in this index. The advantage of building the index over the suspicious document is, that it can be reused over several candidate documents.

Essentially, the Smith-Waterman algorithm should detect the lines in such a dotplot, while being flexible enough to tolerate small perturbations in the lines caused by replaced, inserted or deleted words. Other error tolerant line-detection algorithms like RANSAC [2] are capable to find such lines as well, but still require several iterations over the data during the computation.

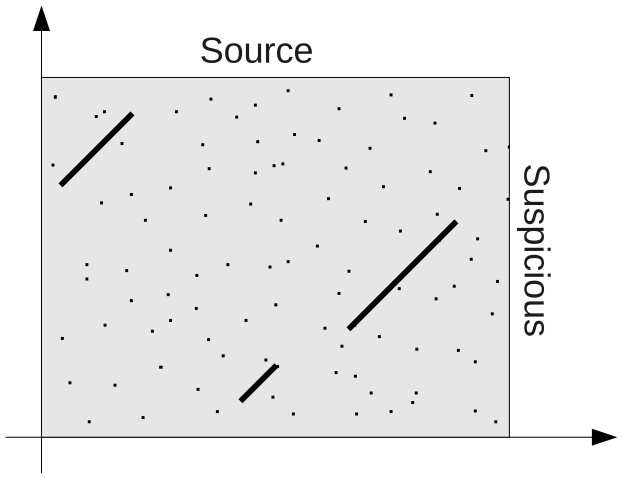

Figure 1. A dotplot plots the positions of terms from a candidate document against their positions in the suspicious document.

These algorithms, however, provide more flexibility than actually needed. Effectively, looking for longer common sub-sequences of high similarity corresponds in our case to finding lines in the dotplot which approximately have a 45 degree angle with both axis. Hence, when subdividing the dotplot in stripes as shown in Figure 2, one can expect each of the lines – which correspond to cases of plagiarism – to lie completely in one stripe.

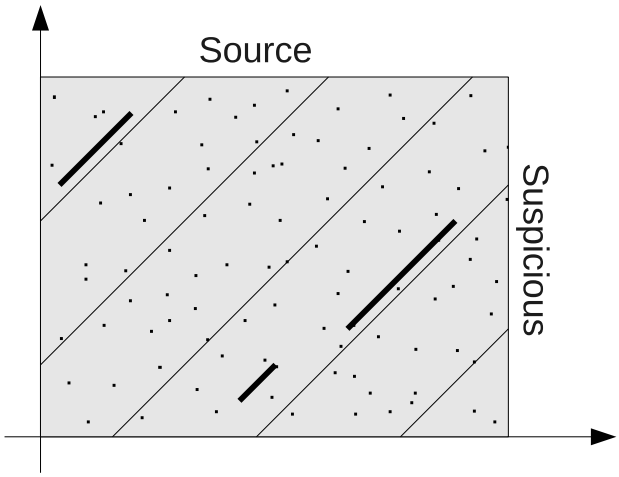

Figure 2. Lines in the dotplot cause by plagiarism lie in stripes of 45 degree angle.

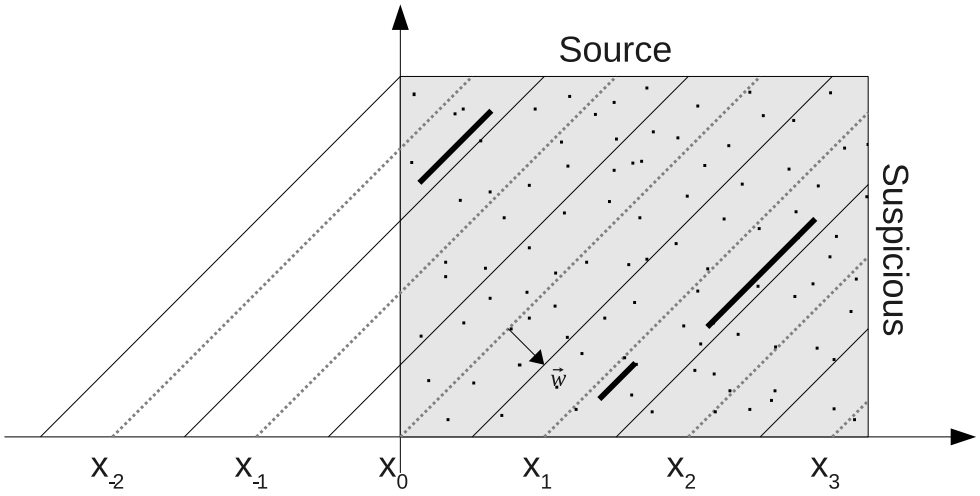

Figure 3. The stripe index for storing token cooccurrences.

Each stripes can be represented by a line x_i in its centre (cf. the dotted lines in Figure 3). These lines can be formulated by using a normal vector $\vec{w}$, that also includes a definition of the width of the stripes:

$$x_i : \langle \vec{p}, \vec{w} \rangle + 2 \cdot i \cdot |\vec{w}| = 0 \tag{2}$$

The advantage of this representation for the stripes is, that for each cooccurrence of a single token in the suspicious and candidate document it is possible to identify immediately, which stripe it falls in. For a cooccurrance at the point (j, k), where j is

the position in the source sequence and k the position in the suspicious sequence, the stripe can be determined by essentially calculating the distance of this point to the line x_0:

$$\textit{stripe}(j,k) = \left\lfloor \frac{1}{-2 \cdot |\vec{w}|} \left\langle \begin{pmatrix} j \\ k \end{pmatrix}, \vec{w} \right\rangle - 0.5 \right\rfloor \tag{3}$$

This lead to the idea, not to compute the dotplot at all, but rather a *stripe-index*. This stripe index stores all cooccurrences of tokens lying in the same stripe as a linked list of entries.

In this way, it is sufficient to consider only those stripes for line detection where a lot of elements are stored in. The stripes with few entries can be skipped as they presumably contain only random cooccurrences. To identify the lines, dots in a stripe are mapped onto the line in its centre, so that in the end, it is enough to consider densities in a one-dimensional data structure.

Lines caused by plagiarism correspond to dense regions in the stripes and are found by comparing the distances between single points on the central stripe lines. If the distance between $n_{\min}$ consecutive points is not larger then $\delta_{\min}$ each, this is considered the start of a dense region, until the next point in the sequence is further away than $\delta_{\min}$.

One issue to take care of was the special case of a line in the dotplot falling exactly on the border between two stripes. In this case and due to the already mentioned possibility of small perturbations in the line, it might happen that some cooccurrences are captured in one stripe and some in the neighbour stripe. This might cause a dense region to be split into smaller parts across two stripes that individually might become too short or not dense enough to be detected. The solution to this problem is to design the stripes to have an overlapping margin. Though, in extreme cases a dense region might still be split in parts, each of them is long enough to be detected individually. One of the post-processing steps described below takes care to join the parts in this case. The improvements obtained in this way depend on the width of the stripes. Given the width eventually chosen for the stripes here, the improvements were marginal.

During the entire process of transforming cooccurrences into points in a stripe, the original positions of the tokens are kept in mind. In this way it is possible to reconstruct for a dense region, what are the corresponding first and last token in the original suspicious and candidate document respectively.

The sub-sequences detected in this way are already a detected region of plagiarism and are resolved into character positions. Before reporting these regions, however, they undergo a few steps of post-processing.

2.3 Post-Processing

Post-processing in the current system consists of three steps:

1. The first step checks for too short regions. As it might happen that very short similar sub-sequences occur by chance, each region that consists of less than a certain number of characters is dropped.

2. The second step merges close regions. If regions lie close to each other or even overlap, they are combined into one region. This happens if the distance between the first and last character of two regions is lower than a threshold.
3. Last, when dealing with a translated document, the positions in the translated English version need to be mapped back to the positions of the original text. This is done via a simple linear interpolation based on the positions of the corresponding chunks of text created during the initial translation process.

3 Evaluation

As the system contains a lot of parameters, a subset of the first 2,000 suspicious documents of the PAN-09 [14] corpus was used to optimise performance. Additionally, the pre-selection process, the detailed analysis and the post processing step were technically split up and intermediary results were serialised to a simple file based representation. This allows individual analysis and evaluation of the steps.

The pre-selection was analysed on how well the Lucene approach is capable of finding good candidates. The gold-standard candidates from the training corpus were used to compute recall-based measures. As the quality of the ranking based on the candidate scores $c(s_i, d_n)$ is of interest as well, recall@10, @20, and @30 as well as the recall over all selected candidates were determined. It turned out that the overall recall differed little from recall@30. So, it seems sufficient to consider only the first 30 candidate documents for a detailed analysis. The recall obtained with the Lucene index was around 0.7. The length l_d of the parts for indexing a document and l_q for searching the index seemed to have little influence on this values. In the end, l_d was set to 5,000 terms and l_q to 50 terms. The threshold θ had been set to 0.5 and retrieval was limited to the top-10 documents for each query.

In the phase of the detailed analysis there are other parameters to tune. The width of the stripes in the stripe-index was set to an equivalent of 50 token, with the stripes having an overlapping margin of 20%. A dense region consisted of at least 5 token with a distance of no more than the equivalent of 10 token.

The post-processing phase used settings chosen according to results reported in previous years [3]. Plagiarised sequences of less than 175 characters were considered random cooccurrences and were discarded. Sequences with a distance of less than 500 characters were merged into one plagiarised content covering also the characters in between.

The performance of this setup can be seen in Table 1. On the training corpus recall and precision obtain quite balanced values and granularity is acceptable. The individual measures as well as the overall score does not reach the best-performing system of last year. But, as the system is a relatively quickly drafted prototype, we did not expect to beat those results.

For comparison, the table also includes the performance of the shortly mentioned approach based on the localized Smith-Waterman algorithm mentioned above. The performance is much lower under all aspects. However, it remains to be said, that this approach was not followed intensively and that a better performance can be expected when tuning the parameters.

Table 1. Evaluation of the system on Training and Test corpus

Corpus	Detailed Analysis	Recall	Precision	Granularity	Overall Score
PAN-09	localized Smith-Waterman	0.2354	0.0789	3.2941	0.0562
PAN-09	Stripe-Index	0.4689	0.4027	1.1402	0.3947
PAN-10	Stripe-Index	0.3174	0.5059	1.8701	0.2564

The performance on the PAN-10 test corpus was different, though. Here recall was lower while precision was higher. The reason for the lower recall can be explained with the lack of an intrinsic analysis component. Evaluating the results of the detection solely against the external cases of plagiarism, recall lies at 0.3905 and, thus, much closer to the values obtain on the training corpus.

Surprising was the increase in granularity, which clearly contributes to the overall lower performance of 0.2564 compared to 0.3947 on the training data. New obfuscation strategies and potentially a different distribution and density of the plagiarised contents could be an explanation. However, this unexpected behaviour remains to be investigated in detail.

Besides the evaluation of effectiveness, also efficiency and run-time performance was of interested. The system was run on commodity hardware: an Lenovo notebook computer with a 2.67 GHz Core i7 processor and 4GB RAM. So far, the system is single threaded and does not make use of parallel execution on several processing cores.

Building the Lucene index for the PAN-10 corpus is negligible and took less than 7 minutes. The pre-selection of candidates was the longest part in the process and required about 28 hours and 21 minutes. The detailed analysis and post-processing was much faster and took a total of 10 hours and 4 minutes. This leads to a total run-time of 38 hours and 32 minutes. Exploiting the inherit potential of parallelising each step of the process, a speedup factor closely correlated to the number of parallel threads can be expected.

4 Conclusion

The system presented in this paper makes two main contributions. First, it uses and evaluates a Lucene index for the candidate selection process. Though, the idea of using standard search systems for plagiarism detection is not new, it had not been applied or evaluated by any system in the PAN plagiarism detection competition of last year. Further, and unlike special purpose indexes that are used in other systems [6,13,1,9,3,10], this allows to easily adapt the system to search the web for candidate documents via web search engines. Second, with the stripe-index used in the detailed analysis step, we obtained good run-time improvements in detecting similar sub-sequences from two documents, compared to the Smith-Waterman and the RANSAC approach.

As the system is in an early prototype stage, the results are far from being competitive with other systems. However, there are several components that offer a lot of potential for improvements:

- Post-processing is working solely on the position and length information of plagiarised parts. No information from the real texts is used. Incorporating the actual

contents of the documents into this process will allow a more elaborated analysis and better decisions on when to discard or merge plagiarised parts.
- The dotplot was built on sorted term 5-grams. Choosing different tokens as base for the dotplot – especially more fine-grained structures – will lead to a higher recall. It remains to analyse how the trade-off in precision and granularity can be counterbalanced, either by improving the line detection method, or by improving the post-processing steps.
- The alignment of positions between the translated and original documents was done in a very simple way. A more fine-grained resolution will improve accuracy for cross-lingual plagiarism detection.
- Semantic changes, like replacing words with synonyms, have not been considered at all so far. They might be incorporated into the inverted index structure over the token positions in the suspicious document.
- Another interesting question could be to replace the vector space model underlying Lucene with a different standard IR model. Given long texts and queries, a promising approach could be to look into retrieval based on language models.
- Finally and as mentioned above, the system can be expected to be sped up also on commodity hardware by implementing thread-based parallelisation.

Overall, as several components so far were designed to provide merely basic functionality, the system can benefit from several improvements, which might contribute to a far better performance in future competitions.

A Acknowledgements

Some of the initial ideas leading to the system described in this paper, especially considering candidate selection based on standard IR techniques, were developed and discussed with students of a course given at the University of Mainz. Hence, I would like to thank Ricard Anufriev, Christian Auth, Florian Feyand, Willy-Roland Monkam, Christian Sigel and Florian Sturm for their contribution in this course.

References

1. Basile, C., Benedetto, D., Caglioti, E., Cristadoro, G., Degli Esposti, M.: A plagiarism detection procedure in three steps: Selection, matches and "squares". In: Proceedings of the SEPLN'09 Workshop on Uncovering Plagiarism, Authorship and Social Software Misuse. pp. 19–23 (2009)
2. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM 24(6), 381–395 (1981)
3. Grozea, C., Gehl, C., Popescu, M.: Encoplot: Pairwise sequence matching in linear time applied to plagiarism detection. In: Proceedings of the SEPLN'09 Workshop on Uncovering Plagiarism, Authorship and Social Software Misuse. pp. 10–18 (2009)
4. Hirschberg, D.S.: A linear space algorithm for computing maximal common subsequences. Commun. ACM 18(6), 341–343 (1975)

5. Kasprzak, J., Brandejs, M., Brandejsova, J.: Distributed aspects of the system for discovering similar documents. In: ITA'09: Proceedings of the 3rd International Conference on Internet Technologies and Applications (2009)
6. Kasprzak, J., Brandejs, M., Kripac, M.: Finding plagiarism by evaluating document similarities. In: Proceedings of the SEPLN'09 Workshop on Uncovering Plagiarism, Authorship and Social Software Misuse. pp. 24–28 (2009)
7. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. Doklady Akademii Nauk SSSR 163(4), 845–848 (1965)
8. Maizel, J.V., Lenk, R.P.: Enhanced graphic matrix analysis of nucleic acid and protein sequences. Proceedings of the National Academy of Sciences of the United States of America 78(12), 7665–7669 (1981)
9. Malcolm, J.A., Lane, P.C.R.: Tackling the pan'09 external plagiarism detection corpus with a desktop plagiarism detector. In: Proceedings of the SEPLN'09 Workshop on Uncovering Plagiarism, Authorship and Social Software Misuse. pp. 29–33 (2009)
10. Shcherbinin, V., Butakov, S.: Using microsoft sql server platform for plagiarism detection. In: Proceedings of the SEPLN'09 Workshop on Uncovering Plagiarism, Authorship and Social Software Misuse. pp. 36–37 (2009)
11. Smith, T.F., Waterman, M.S.: Identification of common molecular subsequences. Journal of Molecular Biology 147(1), 195 – 197 (1981)
12. Stein, B., Lipka, N., Meyer zu Eissen, S.: Meta analysis within authorship verification. In: DEXA '08: Proceedings of the 2008 19th International Conference on Database and Expert Systems Application. pp. 34–39. IEEE Computer Society, Washington, DC, USA (2008)
13. Vallés Balaguer, E.: Putting ourselves in sme's shoes: Automatic detection of plagiarism by the wcopyfind tool. In: Proceedings of the SEPLN'09 Workshop on Uncovering Plagiarism, Authorship and Social Software Misuse. pp. 34–35 (2009)
14. Webis at Bauhaus-Universität Weimar, NLEL at Universidad Politécnica de Valencia: PAN Plagiarism Corpus 2009 (PAN-PC-09). http://www.webis.de/research/corpora (2009), Martin Potthast, Andreas Eiselt, Benno Stein, Alberto Barrón-Cedeño, and Paolo Rosso (editors)
15. Zechner, M., Muhr, M., Kern, R., Michael, G.: External and intrinsic plagiarism detection using vector space models. In: Proceedings of the SEPLN'09 Workshop on Uncovering Plagiarism, Authorship and Social Software Misuse. pp. 47–55 (2009)