The Encoplot Similarity Measure for Automatic Detection of Plagiarism – Extended Technical Report

Cristian Grozea[1⋆] and Marius Popescu[2]

[1] Fraunhofer Institute FIRST, Berlin, Germany
[2] University of Bucharest, Romania

Abstract This paper describes the evolution of our method Encoplot for automatic plagiarism detection and the results of the participation to the PAN'11 competition. The main novelties are the introduction of a new similarity measure and of a new ranking method, which cooperate to rank much better the source–suspicious document pairs when selecting the candidates for the detailed analysis phase. We have obtained excellent results in the competition, ranking 1^{st} on the manually paraphrased cases, 2^{nd} overall in the external plagiarism detection task, and getting the best recall on the non-translated corpus.

1 Introduction

Encoplot is an automatic method for plagiarism detection developed by the authors. It was first introduced in the PAN'09 competition, where it has outperformed all other competing methods [4]. It has been since then enhanced, parallelized and used also for detecting the direction of plagiarism [6].

For this year's competition a new similarity measure for the candidate documents retrieval phase has been introduced, aiming to increase the quality of the ranking and implicitly allowing for an increased recall. Apart from the improvement of the recall, this new similarity measure increases the consistency of the encoplot method, the same strategy being now used both in the candidate documents retrieval phase and in the detailed analysis phase, making thus more probable that a correct find in the first phase will not be missed by the second phase and the other way around.

2 Methods

2.1 Encoplot

The complete details of the basic system and optimized C-language code can be found in [4], therefore we just sketch here briefly the method's main ideas and steps, focusing more on the novelties introduced this year.

Encoplot is both the name of the plagiarism detection system we have developed and used, and the name of the pairwise document comparison algorithm at its core. For the sake of completion, we describe these again here in full, marking also the changes to the version described in [4], where optimized C-language code is included.

⋆ corresponding author – cristian.grozea@brainsignals.de

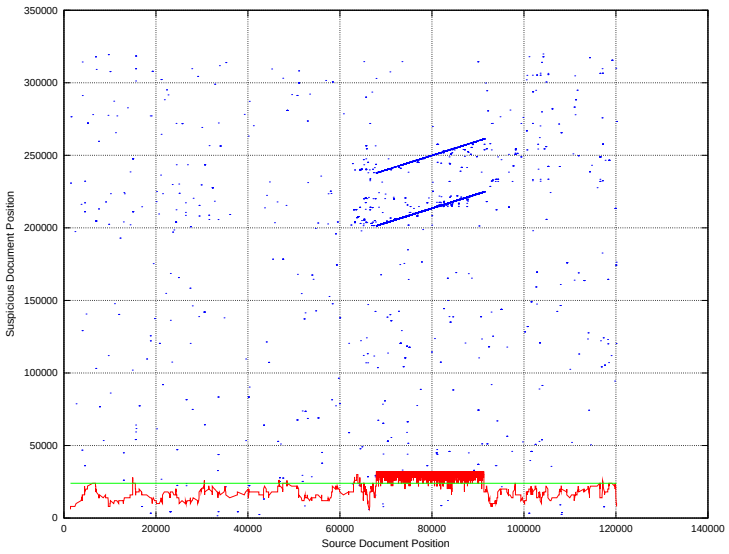

Figure 1. Sample encoplot graph - here the same passage from the source is copied two times into the destination, but not verbatim. The first copy captures most "dots" (N-gram matches).

2.2 The Core Encoplot Algorithm for Comparing Two Documents

Input:

- A and B, two strings of length l_A and respectively l_B over some alphabet Σ.
- N, a natural number – the length of the N-grams.

Output:
$E \subseteq \{1 \dots l_A\} \times \{1 \dots l_B\}$, such that:

- For each $(u, v) \in E$ the N-grams A(u…u+N-1) and B(v…v+N-1) coincide.
- Each index appears at most once: $(u, v) \in E, (u, v') \in E \Rightarrow v = v'$; $(u, v) \in E, (u', v) \in E \Rightarrow u = u'$.
- For each N-gram $w \in \Sigma^N$, let $n_w(A), n_w(B)$ be the number of the occurrences of w in A and B, respectively. Then $|\{(u, v) \in E; A(u \dots u + N - 1) = w\}| = min(n_w(A), n_w(B))$, where $|X|$ denotes the cardinal of the set X. Further more, if $a_w(i)_{i=1 \dots n_w A}$ are the ascendingly ordered positions on which w occurs in A and $b_w(i)_{i=1 \dots n_w B}$ the ones for B, then $(a_w(i), b_w(i)) \in E$, for $i = 1 \dots min(n_w(A), n_w(B))$.

Algorithm:

1. Extract and sort the N-grams in A, let the result be denoted by $S_A \subseteq \{1 \dots l_A\} \times \Sigma^N$. The serial implementation produces a sorting index with a radix sort specialized in sorting N-grams.

2. Do the same for B, let the result be denoted by $S_B \subseteq \{1 \dots l_B\} \times \Sigma^N$.
3. Intersect with a merging procedure the projections of S_A and S_B on their second component, while reporting the values in the first component for the matches.

The outcome of this procedure is a set of pair of indices, the first in each pair being an index in the first document and the second an index in the other document, such that the N-gram starting at the positions indicated by the two indices coincide.

Note that the set produced by encoplot for two documents is a subset of the set produced by the well-known method dotplot [2]. The pairs it produces can be represented graphically as dots, leading to a diagram which is never more cluttered than the one produced by dotplot. For one example, see Figure 1. In contrast to the quadratic order of the running time of dotplot, encoplot has linear running time (as a function of the lengths of the documents). Note as well that no N-gram occurrence is ever matched to more than one occurrence of the N-gram in the other document, and the occurrences without matches in the other document are discarded.

2.3 The Clustering Heuristic

The heuristic employed for clustering the "dots" produced by the encoplot core algorithm into passages consists of the following steps:

1. The dots are projected on the source's axis, local contiguity is declared wherever the distance between two consecutive projected dots is less than 4, then this contiguity is smoothed by convolution with a constant vector of size 16, in order to approximate their density.
2. Within a Monte-Carlo optimization loop (20 attempts), a random starting position is selected, among the projections of the dots on the axis corresponding to the source.
3. This start is treated like the seed of a segment which is extended to the left and to the right as much as possible, while keeping the density of the projections in the segment over a certain limit (50%).
4. If the segment is long enough (512 characters) and the projections within it are dense enough (50%), the dots having projections inside the segment are isolated, their projections on the axis of the suspicious document are pruned of outliers.
5. If the segment on the axis of the source and the segment on the axis of the suspicious document satisfies certain sanity checks (their lengths over 256 and the density of the projections of the dots above 50%), the pair of segments (passages) is selected as a candidate.
6. The best candidate (the one with longest passages found in correspondence) is reported if it satisfies the checks mentioned at the previous step, otherwise the current attempt is labeled as a failure.
7. The dots selected are removed from the set and the Monte-Carlo loop is resumed, up to 30 times. Three consecutive failures to find an acceptable match of passages lead to an early stop of the algorithm.

2.4 The Encoplot Plagiarism Detection System

A run of the whole system consists in the following steps:

1. Preprocess the text, text normalization (including translation when needed).
2. Compute a similarity matrix with one value for each (source document, suspicious document) pair.
3. Rank the document pairs based on their similarity.
4. Analyze the highest ranked pairs in detail, with the following sub-steps:
   - Apply the core encoplot algorithm described above on the two texts and obtain the encoplot data.
   - Cluster the matches of N-grams into matches of passages, using the heuristic algorithm described above.

2.5 Changes to the system

For preprocessing the texts, we have first converted the documents from Unicode UTF-8 to LATIN-1 character set. Then we partly translated some of the German and Spanish documents into English using "Google translate". We couldn't translate everything, as we were already too close to the deadline and Google translate kept refusing the requests. Afterwards, the texts were normalized by changing all letters to lowercase and compacting all white-spaces groups to a single space. Translation and text normalization are new for our system.

For selecting the candidate pairs for detailed analysis, we have introduced both a new similarity measure and a new ranking method.

The previous similarity measure was computed with a standard string kernel, the normalized version of a kernel that computed how many distinct N-grams (N=16 characters) each two documents share. The new similarity measure leverages the ideas of core encoplot algorithm, by computing the encoplot set for the pair of documents of which the similarity is evaluated, followed by a projection of it on the source axis, and a counting of in how many positions a moving window of fixed size (256 characters) contains a number of N-grams matches above a fixed threshold (64). This count is taken to be the similarity of the two documents.

Once the similarity measure matrix is obtained (11093x11093 in this case), we need to rank the pairs based on those values. In the previous years we have considered and contrasted ranking all sources for a given suspicious document and ranking all suspicious documents for a given source. Which one is best depends much on the specific dataset ([5]), therefore we have decided to combine the two rankings in a single ranking that guarantees that whichever pair was selected by either one of those, will be selected by the combined ranking as well. For this we built the min ranking $r_{min}(pair) = min(r_{sources}(pair), r_{suspicious}(pair))$, where $r_{sources}(pair)$ is the rank of the pair in its column in the similarity matrix and $r_{suspicious}(pair)$ is its rank in the row containing it in the same matrix.

The detailed analysis remained almost unchanged, still we have had to do a change related to a serious problem the 2010 competition corpus had, namely some of the passages from the source were copied multiple times into the destination suspicious document – a substantial amount: out of 55723 external plagiarism instances, 10694 ($> 19\%$) had the multiplicity at least 2, 3483 multiplicity at least 3. The maximum multiplicity of a single passage was 17 (!).

This probably explains our suspiciously low recall in the 2010 competition on the non-obfuscated cases (and other subcorpora). As a side effect of the speed and space optimizations the core encoplot algorithm offers over dotplot, for the simple case when there is no obfuscation at all and just verbatim copying multiple times, only the first copy of a passage is matched. For the example in Figure 1 the copying isn't verbatim, therefore a small proportion of the matches are shifted to the second copy. To understand why, remember that each position in the source is paired with at most one position in the suspicious document. Therefore a full match of the source passage fully "consumes" it, and it cannot match any of the subsequent copies. Having a second copy of the same passage in the source would allow for a second match and so on. To cope with that, we have concatenated each source with itself 4 times before analyzing the pair in detail with our heuristic, creating thus 4 copies in the source of each passage previously there. The number 4 has been chosen as a compromise in order to balance the effort and the expected increase in recall. This change has had the undesirable effect of increasing the running time and has raised thus the sparsity requirements for the candidate selection procedure.

2.6 Implementation Details

Software infrastructure We have used C, C++, Perl, Shell scripts and Octave under Linux.

For parallelizing the computation of the similarity matrix we have switched from OpenMP to the framework StarSs [8], developed at the Barcelona Supercomputing Center (BSC). The precise implementation we have used was SMPSs [7], meant for multi-core machines. The advantage of this task-based parallelization framework is that with simple annotations, tasks can be defined that are executed then in parallel on different execution threads. The necessary tasks dependency detection, locking and thread-to-thread communication and data transfers are performed automatically by the framework. We have developed this version of the code during a HPC Europa2 virtual visit to the BSC, where the first author had the chance to work together on this with Jesus Labarta, Rosa M. Badia and Aislan Gomide Foina and to run the code on machines with up to 256 cores (article in preparation).

Hardware We have used three systems, not simultaneously:

- A 12-core machine: 2xAMD 6-core Opteron 2427 64bit 2200 MHz in our multi-core lab. Unfortunately it became defective before we finished the analysis of the data.
- A Condor [12] cluster: up to 34 cores (shared) AMD Opteron 275 at 2200MHz
- A 4-core machine: 1x4-core Intel Xeon E5540 at 2.53 GHz, as a replacement for the no longer available 12-core machine.

Implementation tricks/techniques The text normalization used in the preprocessing phase, as well as the translation require matching the positions in the transformed documents back into the original texts. For this we have kept index files, which mapped

each position in each transformed text to the corresponding position in the original text. Our first implementations were either too slow or using too much memory in the lookup phase, therefore we have optimized the implementation by choosing to keep the indices in fixed record length binary format files and consulting those directly on the disk, using the function "seek", without explicitly loading the whole content of the mapping files into the memory, and letting thus the Linux operating system employ optimally the various buffers and caches.

Processing time Translation was prohibitively slow, and we have not have enough time for it (just 48h). Other preprocessing took less than 1 hour on the 12-core machine. The computation of the similarity matrix took 24h on the same machine. Ranking takes a few minutes. The detailed analysis would have taken 3 full days on the 12 core machine therefore we have moved it on the cluster, where it took 24h. We chose not to use for translating the extra week offered as prolongation – after we have submitted – as we are interested more in the pure external plagiarism detection and consider the translation a separate problem (see the Discussion section for more).

3 Evaluation

3.1 Dataset

This year's competition test data consisted of 11093 source documents in English, German (about 500) and Spanish (about 200).

There were 49621 plagiarism cases (passages copied with or without translation/obfuscation). The distribution of their types is given in Table 1.

Table 1. The structure of the competition dataset - plagiarism cases

Category	Subcategory	Count
Entire set		49,621
Simulated		4,609
Artificial		39,870
	no obfuscation	**976**
	low obfuscation	19,779
	high obfuscation	19,115
Translated		5,142
	by hand	433
	automatic	4,709

For our pairwise document matching approach it is interesting to note that this amounts to more than 120 million document pairs to investigate and requires thus very efficient implementation. At document level, out of all these pairs only 17674 contained plagiarism cases.

3.2 Analysis and Results

As our method consists of two distinct phases, ranking the document pairs followed by detailed analysis of the selected ones, it is useful to measure the performance in each phase, in order to be able to contrast the alternative approaches.

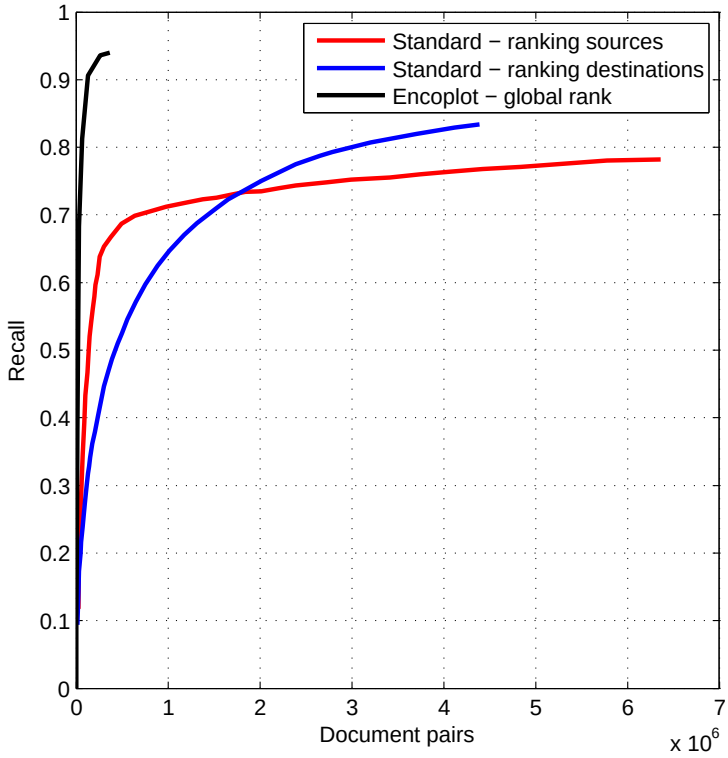

Figure 2. Comparative recall at document level on the 2010 competition corpus of the newly introduced encoplot similarity measure and of the two ranking variants we have employed in 2010, based on a standard string kernel. The encoplot similarity measure graph ends when all document pairs with non-null similarity values are included.

The purpose of the ranking is to identify the pairs of documents for which there is a high chance that there is at least one plagiated passage from the source to the suspicious document. An ideal ranking should put first the pairs where such a relation exists and last the other pairs. A good ranking should be an approximation of this. It should enable the achievement of a good recall at document level without having to select too many document pairs. Note that all rankings (even a random one) will achieve a recall of 100%, the latest after selecting all pairs. What is important is how fast the recall is increasing when the selection is extended to include more and more of the document pairs, as ranked.

For examining the performance of the ranking, we have considered two types of graphs: effort versus recall – where the recall is shown together with the number of pairs needed to achieve that recall – and a standard precision versus recall.

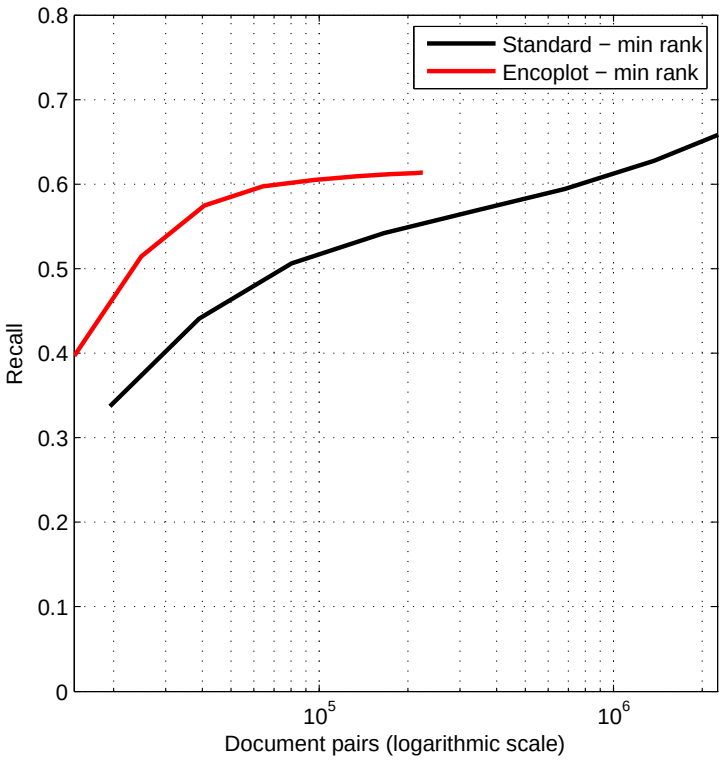

Figure 3. Comparative recall at document level on the 2011 competition corpus of the encoplot similarity measure and of the best ranking variant for the standard string kernel. The encoplot similarity measure graph ends when all document pairs with non-null similarity values are included.

On the 2010 competition data, the effect of the new similarity ranking is impressive. Both on the "effort versus recall" plots (Figure 2) and in the "precision versus recall" plots (Figure 4), the new similarity method outperforms the standard one consistently by a great margin.

On the 2011 competition data, while it still dominates in the "affordable effort" range, it is eventually outperformed by the standard kernel coupled with min rank, Figure 3.

3.3 Results

The results on the 2011 competition data are summarized in Table 2.

Our team has ranked 2nd on the external plagiarism detection task.

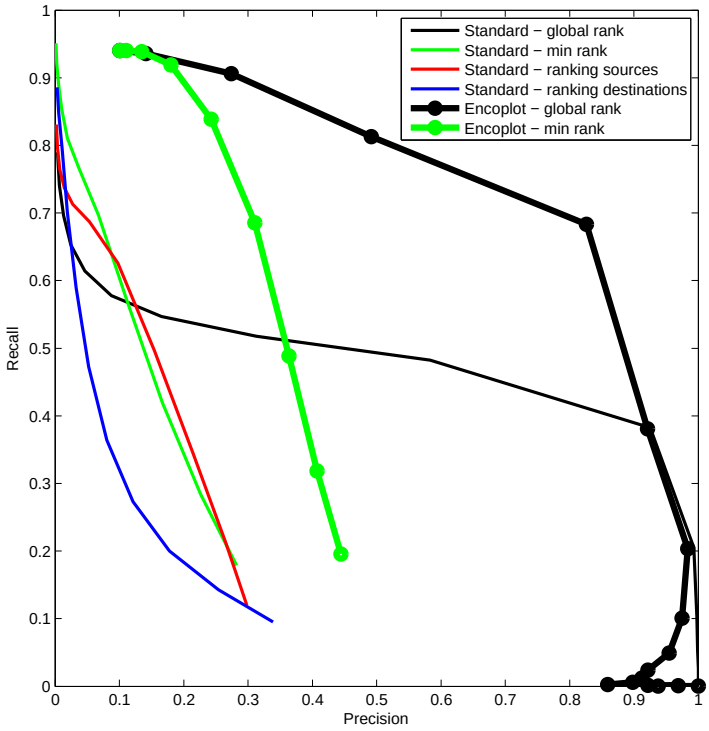

Figure 4. Precision-Recall plots for several ranking types for the standard and the encoplot based similarity measures.

We have had the best score among all teams on the category that mimics best the real human plagiarism: manual paraphrasing. Also we have had the best recall values on manual paraphrasing and on both subcategories of automatic paraphrasing.

By using the annotations provided we have been able to compute for all teams the recall on the non-translated cases subcorpus. We lead also there, with 0.3512, followed by the team that ranked first with 0.3468. It was impossible to compute the precision on the same subcorpus without having access to the answer files of the other teams.

We have tested the new method also on the 2010 competition data, on which it had a plagdet score 0.72, with recall 66% and precision 86%. These results are very encouraging, they would have ranked our method on the second place, without even handling the translated cases (14%) - whereas the team which ranked first in 2010 did handle those too.

Table 2. Results on 2011 Competition Data

Subset	Recall	Precision	F-score	Granularity	Plagdet score
Entire corpus	0.34	0.81	0.48	1.22	0.42
No paraphrasing	0.90	0.84	0.86	1.02	0.85
Manual paraphrasing	**0.36**	**0.96**	**0.53**	1.06	**0.50**
Automatic paraphrasing low obfuscation	**0.58**	0.90	0.71	1.27	0.60
Automatic paraphrasing high obfuscation	**0.08**	0.64	0.14	1.19	0.13
Manual translation	0.08	0.25	0.12	1.01	0.12
Automatic translation	0.23	0.40	0.29	1.07	0.28

4 Discussion

The distribution of the plagiarism types in this year's test data is far from matching their distribution in the training corpus, PAN-PC-10 [10]. Most notably, the amount of passages copied without changes (no obfuscation, no translation) was heavily reduced, from 40% to less than 2%. Therefore, tuning on the training corpus could have been to the disadvantage of the participants. We were consistent with our previous approach of not tuning our method to a particular and still mostly artificial training corpus. This may explain to some extent our top performance in the cases of human plagiarism, which is indeed the only one of practical interest. We expect that the next corpora will contain more and more such human and manually simulated plagiarism to the detriment of the artificially generated using questionable choices (see the repeated insertion of the same passage from one source into a single destination suspicious document mentioned above).

4.1 The Issue with the Translation

In our view, automatic translation is a rather separate problem in NLP. Of course, there could exist an approach to cross-language plagiarism detection that doesn't bring first all texts to the same language by automatic translation, but we are not aware of any such approach with good performance either in the competitions so far, or in the existing literature. The idea to translate first, followed by the same language plagiarism detection is neither a scientific contribution, nor a distinguishing feature for a plagiarism detection system.

Assuming thus that everyone tackling the cross-language plagiarism detection uses automatic translation, the choice and the flaws of the later bring randomness into the evaluation of the pure plagiarism detection methods. For example, using the very same translation engine employed when building the corpus does artificially inflate the scores of the teams who happened to employ the same engine.

As most of the translation used in the dataset was automatic, even when a competitor would have had the resources to use high quality human translation of all the sources that were not in English (a daunting task!), the resulting mismatch between this translation and the fairly poor automatic translation will resemble a heavy obfuscation, making the actual plagiarism detection task only more difficult.

We have tested several translation engines for the same language, and their (very frequent) translation mistakes rarely matched.

The dependence on online translation services with unknown future availability is not a good design idea. As already known, Google has announced closing down its online translation service. Looking for alternatives, we have tested two desktop translation engines, OpenLogos [11] for German and Apertium [3] for Spanish. Unfortunately they were slow and not very accurate.

A telling example of how much increase in obfuscation is to be expected when mismatching the translation engine can be seen by examining alternative translations for the first paragraph of source document 10796:

"Mit Rücksicht auf den von uns klar erkannten Charakter des Originales des Papyrus als eines sehr ungenauen Collegienheftes, dessen Rechnungen ebensosehr wie die vorkommenden Zeichnungen von der Mittelmässigkeit seines Zusammenstellers beredtes Zeugniss ablegen, zweifeln wir keinen Augenblick, dass die fragliche Figur ein rechtwinkliges Dreieck vorzustellen hatte."

Openlogos: "Taking into consideration the character of the original clearly recognized by us of papyrus as a very inexact Collegienheftes whose calculations just as much as the occurring drawings of the Mittelmässigkeit of its/his Zusammenstellers beredtes Zeugniss take off we doubt no moment that the figure in question had to present a rectangular triangle."

Google translate: "With regard to the best of our clearly recognized character of the original of the papyrus as a very inaccurate Colleges issue, put the bills as much as occur drawings of the mediocrity of his co-Tellers eloquent testimony, we doubt for a moment that the relevant figure was imagining a right triangle."

Yahoo Babelfish: "Out of consideration on by us clearly recognized the character of the original of the papyrus as a very inaccurate Collegienheftes, its calculations likewise very much like the occurring designs of the mediocrity of its Zusammenstellers eloquent things-eat placings, doubt we no instant that the questionable figure had to present a right-angled triangle."

4.2 The issue with the Granularity correction in the Plagdet score

The granularity has been introduced for plagiarism detection in [9]. It was meant to correct the standard F-score for excessive splitting of the plagiarized passages retrieved. It is an ad-hoc correction that divides the F-score by $log_2(1+granularity)$. In our view, it exhibits unwanted behavior in certain cases. For example, let's assume we compare with plagdet two methods, one having recall 33.33%, precision 100% and granularity 1 with another method having both precision and recall 100% and granularity 3. The two methods will obtain the very same plagdet score, 0.5, as a result of applying the granularity correction, although the second method is obviously to be preferred. It has 100% recall and precision, it finds everything and nothing more and even the splitting is very far from excessive. No user will ever prefer a software that fails to find two thirds of the cases to a software that finds them all and even displays each as one block (when colouring text blocks, the adjacent parts will visually join).

More thought should be spent in finding a reasonable plagiarism detection score.

4.3 Interesting Examples

As part of our analysis we have searched for examples where the similarity measure malfunctioned. One such example is (source 4314, suspicious 202) for which similarity value computed was among the top ones, although the pair should contain no plagiarism cases.

We have found that both texts are authored by Thomas Aquinas and seem to be both part of "Summa Theologica", sharing the topics and exhibiting a very constrained and formal language. The detailed analysis found no matches between the two documents, but it's still interesting that our similarity measure captured the link between those documents.

5 Conclusion

In this paper the newest changes to the plagiarism detection method Encoplot have been presented, as well as the results in the competition PAN'11. The main change was replacing the standard, kernel based pairwise document similarity measure by a new similarity measure that includes some of the encoplot core ideas. We have shown that this new similarity measure is to be preferred when only a small part of the whole set of document pairs can be analyzed in detail, as the recall at document level increases much faster with the here proposed similarity measure. The results in the competition were excellent, our team obtained the best scores for the manual paraphrasing subcorpus and ranked 2^{nd} overall on the external plagiarism detection task, obtaining also the best recall on the subcorpus of non-translated plagiarism cases.

Acknowledgment: The parallelization of Encoplot has been performed under the HPC-EUROPA2 project (project number: 228398) with the support of the European Commission - Capacities Area - Research Infrastructures.

References

1. Braschler, M., Harman, D., Pianta, E. (eds.): CLEF 2010 LABs and Workshops, Notebook Papers, 22-23 September 2010, Padua, Italy (2010)
2. Clough, P.: Old and new challenges in automatic plagiarism detection. National Plagiarism Advisory Service (2003)
3. Forcada, M., Tyers, F., Ramírez-Sánchez, G.: The Apertium machine translation platform: five years on. In: Proceedings of the First International Workshop on Free/Open-Source Rule-Based Machine Translation. pp. 3–10. Citeseer (2009)
4. Grozea, C., Gehl, C., Popescu, M.: ENCOPLOT: Pairwise Sequence Matching in Linear Time Applied to Plagiarism Detection. In: 3rd PAN WORKSHOP. UNCOVERING PLAGIARISM, AUTHORSHIP AND SOCIAL SOFTWARE MISUSE. p. 10 (2009)
5. Grozea, C., Popescu, M.: Encoplot - Performance in the Second International Plagiarism Detection Challenge - Lab Report for PAN at CLEF 2010 . In: Braschler et al. [1]
6. Grozea, C., Popescu, M.: Who's the Thief? Automatic Detection of the Direction of Plagiarism. In: Gelbukh, A.F. (ed.) CICLing. Lecture Notes in Computer Science, vol. 6008, pp. 700–710. Springer (2010)

7. Perez, J., Badia, R., Labarta, J.: A dependency-aware task-based programming environment for multi-core architectures. In: Cluster Computing, 2008 IEEE International Conference on. pp. 142–151 (2008)
8. Planas, J., Badia, R.M., Ayguadé, E., Labarta, J.: Hierarchical task based programming with StarSs. International Journal of High Performance Computing 23(3), 284–299 (August 2009)
9. Potthast, M., Stein, B., Barrón-Cedeño, A., Rosso, P.: An evaluation framework for plagiarism detection. In: Proceedings of the 23rd International Conference on Computational Linguistics: Posters. pp. 997–1005. Association for Computational Linguistics (2010)
10. Potthast, M., Barrón-Cedeño, A., Eiselt, A., Stein, B., Rosso, P.: Overview of the 2nd International Competition on Plagiarism Detection. In: Braschler et al. [1]
11. Scott, B., Barreiro, A.: OpenLogos MT and the SAL representation language (2009)
12. Thain, D., Tannenbaum, T., Livny, M.: Distributed computing in practice: the condor experience. Concurrency - Practice and Experience 17(2-4), 323–356 (2005)