3rd PAN WORKSHOP. UNCOVERING PLAGIARISM, AUTHORSHIP AND SOCIAL SOFTWARE MISUSE

25th ANNUAL CONFERENCE OF THE SPANISH SOCIETY FOR NATURAL LANGUAGE PROCESSING, SEPLN 2009

Benno Stein
Paolo Rosso
Efstathios Stamatatos
Moshe Koppel
Eneko Agirre
(eds.)

UNIVERSIDAD POLITÉCNICA DE VALENCIA
EDITORIAL UPV Ref.:2009.2086

Preface

The PAN workshop brought together experts and researchers around the exciting and future-oriented topics of plagiarism detection, authorship identification, and the detection of social software misuse. The development of new solutions for these problems can benefit from the combination of existing technologies, and in this sense the workshop provides a platform that spans different views and approaches.

Plagiarism analysis is a collective term for computer-based methods to identify a plagiarism offense. In connection with text documents we distinguish between corpus-based and intrinsic analysis: the former compares suspicious documents against a set of potential original documents, the latter identifies potentially plagiarized passages by analyzing the suspicious document with respect to changes in writing style.

Authorship identification divides into so-called attribution and verification problems. In the authorship attribution problem, one is given examples of the writing of a number of authors and is asked to determine which of them authored given anonymous texts. In the authorship verification problem, one is given examples of the writing of a single author and is asked to determine if given texts were or were not written by this author. As a categorization problem, verification is significantly more difficult than attribution. Authorship verification and intrinsic plagiarism analysis represent two sides of the same coin.

"Social Software Misuse" can nowadays be noticed on many social software based platforms. These platforms like Blogs, sharing sites for photos and videos, wikis and on-line forums are contributing up to one third of new Web content. "Social Software Misuse" is a collective term for anti-social behavior in on-line communities; an example is the distribution of spam via the e-mail infrastructure. Interestingly, spam is one of the few misuses for which detection technology is developed at all, though various forms of misuse exist that threaten the different on-line communities. Our workshop shall close this gap and invites contributions concerned with all kinds of social software misuse.

September 2009

Benno Stein
Paolo Rosso
Efstathios Stamatatos
Moshe Koppel
Eneko Agirre

Organization

PAN 2009 was organized by the Web Technology & Information Systems Group of the Bauhaus University Weimar, the Natural Language Engineering Lab. of the Technical University of Valencia, the Dept. of Information and Communication Systems Engineering of the University of the Aegean, the Department of Computer Science of the Bar-Ilan University and the IXA Research Group of the Basque Country University.

Organizing Committee

Benno Stein	Bauhaus University Weimar
Paolo Rosso	Technical University of Valencia
Efstathios Stamatatos	University of the Aegean
Moshe Koppel	Bar-Ilan University
Eneko Agirre	University of the Basque Country

Program Committee

Alberto Barrón-Cedeño	Technical University of Valencia
Benno Stein	Bauhaus University Weimar
Carole Chaski	Institute for Linguistic Evidence
Christian Guetl	University of Technology Graz
David Pinto	B. Autonomous University of Puebla
Efstathios Stamatatos	University of the Aegean
Eneko Agirre	University of the Basque Country
Fazli Can	Bilkent University
George Mikros	National and Capodestrian University of Athens
Graeme Hirst	University of Toronto
Martin Potthast	Bauhaus University Weimar
Mike Reddy	Newport Business School
Moshe Koppel	Bar-Ilan University
Ozlem Uzuner	State University of New York
Paolo Rosso	Technical University of Valencia
Paul Clough	University of Sheffield
Shlomo Argamon	Illinois Institute of Technology
Sven Meyer zu Eissen	Bayer Business Services

1st International Competition on Plagiarism Detection

Martin Potthast	Bauhaus University Weimar
Andreas Eiselt	Bauhaus University Weimar
Alberto Barrón-Cedeño	Technical University of Valencia

Workshop and Competition Sponsors

Table of Contents

Overview of the 1st International Competition on Plagiarism Detection*

Martin Potthast Benno Stein Andreas Eiselt
Web Technology & Information Systems Group
Bauhaus-Universität Weimar
<first name>.<last name>@uni-weimar.de

Alberto Barrón-Cedeño Paolo Rosso
Natural Language Engineering Lab, ELiRF
Universidad Politécnica de Valencia
{lbarron|prosso}@dsic.upv.es

Abstract: The 1st International Competition on Plagiarism Detection, held in conjunction with the 3rd PAN workshop on Uncovering Plagiarism, Authorship, and Social Software Misuse, brought together researchers from many disciplines around the exciting retrieval task of automatic plagiarism detection. The competition was divided into the subtasks external plagiarism detection and intrinsic plagiarism detection, which were tackled by 13 participating groups.

An important by-product of the competition is an evaluation framework for plagiarism detection, which consists of a large-scale plagiarism corpus and detection quality measures. The framework may serve as a unified test environment to compare future plagiarism detection research. In this paper we describe the corpus design and the quality measures, survey the detection approaches developed by the participants, and compile the achieved performance results of the competitors.

Keywords: Plagiarism Detection, Competition, Evaluation Framework

1 Introduction

Plagiarism and its automatic retrieval have attracted considerable attention from research and industry: various papers have been published on the topic, and many commercial software systems are being developed. However, when asked to name the best algorithm or the best system for plagiarism detection, hardly any evidence can be found to make an educated guess among the alternatives. One reason for this is that the research field of plagiarism detection lacks a controlled evaluation environment. This leads researchers to devise their own experimentation and methodologies, which are often not reproducible or comparable across papers. Furterhmore, it is unknown which detection quality can at least be expected from a plagiarism detection system.

To close this gap we have organized an international competition on plagiarism detection. We have set up, presumably for the first time, a controlled evaluation environment for plagiarism detection which consists of a large-scale corpus of artificial plagiarism and detection quality measures. In what follows we overview the corpus, the quality measures, the participants' detection approaches, and their obtained results.

1.1 Related Work

Research on plagiarism detection has been surveyed by Maurer, Kappe, and Zaka (2006) and Clough (2003). Particularly the latter provides well thought-out insights into, even today, "[...] *new challenges in automatic plagiarism detection*", among which the need for a standardized evaluation framework is already mentioned.

With respect to the evaluation of commercial plagiarism detection systems, Weber-Wulff and Köhler (2008) have conducted a manual evaluation: 31 handmade cases of plagiarism were submitted to 19 systems. The sources for the plagiarism cases were selected from the Web and the systems were judged by their capability to retrieve them. Due to the use of the Web, the experiment is not controlled which limits reproducibility, and since each case is only about two pages long there are concerns with respect to the study's representativeness. However, com-

*http://www.webis.de/research/workshopseries/pan-09

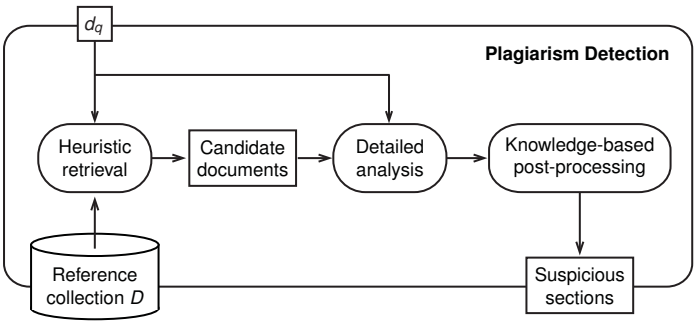

Figure 1: Generic retrieval process for external plagiarism detection.

mercial systems are usually not available for a close inspection which may leave no other choice to evaluate them.

1.2 Plagiarism Detection

The literature on the subject often puts plagiarism detection on a level with the identification of highly similar sections in texts or other objects. But this does not show the whole picture. From our point of view plagiarism detection divides into two major problem classes, namely external plagiarism detection and intrinsic plagiarism detection. Both of which include a number of subproblems and the frequently mentioned step-by-step comparison of two documents is only one of them.

For external plagiarism detection Stein, Meyer zu Eissen, and Potthast (2007) introduce a generic three-step retrieval process. The authors consider that the source of a plagiarism case may be hidden in a large reference collection, as well as that the detection results may not be perfectly accurate. Figure 1 illustrates this retrieval process. In fact, all detection approaches submitted by the competition participants can be explained in terms of these building blocks (cf. Section 4).

The process starts with a suspicious document d_q and a collection D of documents from which d_q's author may have plagiarized. Within a so-called heuristic retrieval step a small number of candidate documents D_x, which are likely to be sources for plagiarism, are retrieved from D. Note that D is usually very large, e.g., in the size of the Web, so that it is impractical to compare d_q one after the other with each document in D. Then, within a so-called detailed analysis step, d_q is compared section-wise with the retrieved candidates. All pairs of sections (s_q, s_x) with $s_q \in d_q$ and $s_x \in d_x$, $d_x \in D_x$, are to be retrieved such that s_q and s_x have a high similarity under some retrieval model. In a knowledge-based post-processing step those sections are filtered for which certain exclusion criteria hold, such as the use of proper citation or literal speech. The remaining suspicious sections are presented to a human, who may decide whether or not a plagiarism offense is given.

Intrinsic plagiarism detection has been studied in detail by Meyer zu Eissen and Stein (2006). In this setting one is given a suspicious document d_q but no reference collection D. Technology that tackles instances of this problem class resembles the human ability to spot potential cases of plagiarism just by reading d_q.

1.3 Competition Agenda

We have set up a large-scale corpus (D_q, D, S) of "artificial plagiarism" cases for the competition, where D_q is a collection of suspicious documents, D is a collection of source documents, and S is the set of annotations of all plagiarism cases between D_q and D. The competition divided into two tasks and into two phases for which the corpus was split up into 4 parts; one part for each combination of tasks and phases. For simplicity the sub-corpora are not denoted by different symbols.

Competition tasks and phases:

- *External Plagiarism Detection Task.* Given D_q and D the task is to identify the sections in D_q which are plagiarized, and their source sections in D.
- *Intrinsic Plagiarism Detection Task.* Given only D_q the task is to identify the plagiarized sections.

- *Training Phase.* Release of a training corpus (D_q, D, S) to allow for the development of a plagiarism detection system.
- *Competition Phase.* Release of a competition corpus (D_q, D) whose plagiarism cases were to be detected and submitted as detection annotations, R.

Participants were allowed to compete in either of the two tasks or both. After the competition phase the participants' detections were evaluated, and the winner of each task as well as an overall winner was determined as that participant whose detections R best matched S in the respective competition corpora.

2 Plagiarism Corpus

The PAN plagiarism corpus, PAN-PC-09, comprises 41 223 text documents in which 94 202 cases of artificial plagiarism have been inserted automatically (Webis at Bauhaus-Universität Weimar and NLEL at Universidad Politécnica de Valencia, 2009). The corpus is based on 22 874 book-length documents from the Project Gutenberg.[1] All documents are, to the best of our knowledge, public domain; therefore the corpus is available free of charge to other researchers. Important parameters of the corpus are the following:

- *Document Length.* 50% of the documents are small (1-10 pages), 35% medium (10-100 pages), and 15% large (100-1000 pages).
- *Suspicious-to-Source Ratio.* 50% of the documents are designated as suspicious documents D_q, and 50% are designated as source documents D (see Figure 2).
- *Plagiarism Percentage.* The percentage θ of plagiarism per suspicious document $d_q \in D_q$ ranges from 0% to 100%, whereas 50% of the suspicious documents contain no plagiarism at all. Figure 3 shows the distribution of the plagiarized documents for the external test corpus. For the intrinsic test corpus applies the hashed part of the distribution.
- *Plagiarism Length.* The length of a plagiarism case is evenly distributed between 50 words and 5000 words.

[1] `http://www.gutenberg.org`

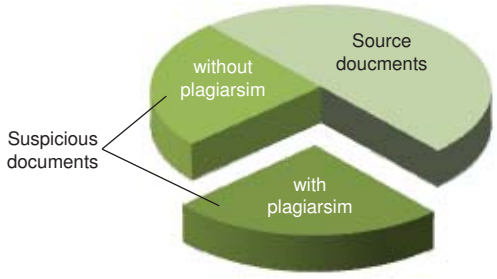

Figure 2: Distribution of suspicious documents (with and without plagiarism) and source documents.

- *Plagiarism Languages.* 90% of the cases are monolingual English plagiarism, the remainder of the cases are cross-lingual plagiarism which were translated automatically from German and Spanish to English.
- *Plagiarism Obfuscation.* The monolingual portion of the plagiarism in the external test corpus was obfuscated (cf. Section 2.1). The degree of obfuscation ranges evenly from none to high.

Note that for the estimation of the parameter distributions one cannot fall back on large case studies on real plagiarism cases. Hence, we decided to construct more simple cases than complex ones, where "simple" refers to short lengths, a small percentage θ, and less obfuscation. However, complex cases are overrepresented to allow for a better judgement whether a system detects them properly.

2.1 Obfuscation Synthesis

Plagiarists often modify or rewrite the sections they copy in order to obfuscate the plagiarism. In this respect, the automatic synthesis of plagiarism obfuscation we applied when constructing the corpus is of particular interest. The respective synthesis task reads

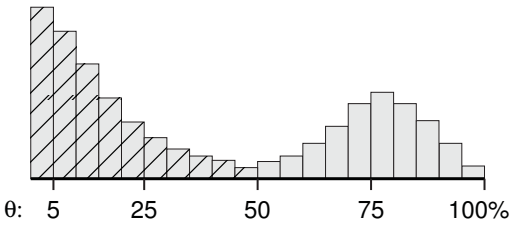

Figure 3: Distribution of the plagiarism percentage θ in the external test corpus. For the intrinsic test corpus applies the hashed part only.

as follows: given a section of text s_x, create a section s_q which has a high content similarity to s_x under some retrieval model but with a (substantially) different wording than s_x.

An optimal obfuscation synthesizer, i.e., an automatic plagiarist, takes an s_x and creates an s_q which is human-readable and which creates the same ideas in mind as s_x does when read by a human. Today, such a synthesizer cannot be constructed. Therefore, we approach the task from the basic understanding of content similarity in information retrieval, namely the bag-of-words model. By allowing our obfuscation synthesizers to construct texts which are not necessarily human-readable they can be greatly simplified. We have set up three heuristics to construct s_q from s_x:

- *Random Text Operations.* Given s_x, s_q is created by shuffling, removing, inserting, or replacing words or short phrases at random. Insertions and replacements are, for instance, taken from the document d_q, the new context of s_q.
- *Semantic Word Variation.* Given s_x, s_q is created by replacing each word by one of its synonyms, antonyms, hyponyms, or hypernyms, chosen at random. A word is retained if neither are available.
- *POS-preserving Word Shuffling.* Given s_x its sequence of parts of speech (POS) is determined. Then, s_q is created by shuffling words at random while the original POS sequence is maintained.

2.2 Critical Remarks

The corpus has been conceived and constructed only just in time for the competition so that there may still be errors in it. For instance, the participants pointed out that there are a number of unintended overlaps between unrelated documents. These accidental similarities do not occur frequently, so that an additional set of annotations solves this problem.

The obfuscation synthesizer based on random text operations produces anomalies in some of the obfuscated texts, such as sequences of punctuation marks and stop words. These issues were not entirely resolved so that it is possible to find some of the plagiarism cases by applying a kind of anomaly detection. Nevertheless, this was not observed during the competition.

Finally, by construction the corpus does not accurately simulate a heuristic retrieval situation in which the Web is used as reference collection. The source documents in the corpus do not resemble the Web appropriately. Note, however, that sampling the Web is also a problem for many ranking evaluation frameworks.

3 Detection Quality Measures

A measure that quantifies the performance of a plagiarism detection algorithm will resemble concepts in terms of precision and recall. However, these concepts cannot be transferred one-to-one from the classical information retrieval situation to plagiarism detection. This section explains the underlying connections and introduces a reasonable measure that accounts for the particularities.

Let d_q be a plagiarized document; d_q defines a sequence of characters each of which is either labeled as plagiarized or non-plagiarized. A plagiarized section s forms a contiguous sequence of plagiarized characters in d_q. The set of all plagiarized sections in d_q is denoted by S, where $\forall s_i, s_j \in S : i \neq j \rightarrow (s_i \cap s_j = \emptyset)$, i.e., the plagiarized sections do not intersect. Likewise, the set of all sections $r \subset d_q$ found by a plagiarism detection algorithm is denoted by R. See Figure 4 for an illustration.

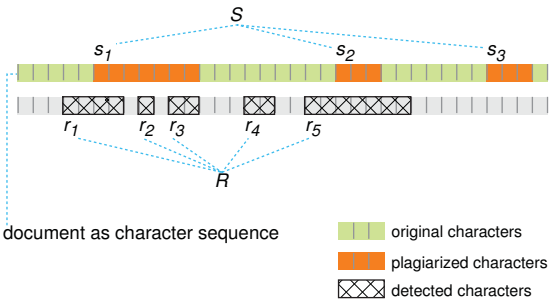

Figure 4: A document as character sequence, including plagiarized sections S and detections R returned by a plagiarism detection algorithm. The figure is drawn at scale $1 : n$ chars, $n \gg 1$.

If the *characters* in d_q are considered as basic retrieval units, precision and recall for a given $\langle d_q, S, R \rangle$ compute straightforwardly. This view may be called *micro-averaged* or system-oriented. For the situation shown in Figure 4 the micro-averaged precision is 8/16, likewise, the micro-averaged recall is 8/13. The advantage of a micro-averaged view is its clear computational semantics, which comes

at a price: given an imbalance in the lengths of the elements in S—which usually correlates with the detection difficulty of a plagiarized section—the explanatory power of the computed measures is limited.

It is more natural to treat the contiguous sequences of plagiarized characters as basic retrieval units. In this sense each $s_i \in S$ defines a query q_i for which a plagiarism detection algorithm returns a result set $R_i \subseteq R$. This view may be called *macro-averaged* or user-oriented. The recall of a plagiarism detection algorithm, rec_{PDA}, is then defined as the mean of the returned fractions of the plagiarized sections, averaged over all sections in S:

$$rec_{PDA}(S, R) = \frac{1}{|S|} \sum_{s \in S} \frac{|s \sqcap \bigcup_{r \in R} r|}{|s|}, \quad (1)$$

where $\sqcap$ computes the positionally overlapping characters.

Problem 1. The *precision* of a plagiarism detection algorithm is not defined under the macro-averaged view, which is rooted in the fact that a detection algorithm does not return a unique result set for each plagiarized section $s \in S$. This deficit can be resolved by switching the reference basis. Instead of the plagiarized sections, S, the algorithmically determined sections, R, become the targets: the precision with which the queries in S are answered is identified with the recall of R under S.[2] By computing the mean average over the $r \in R$ one obtains a definite computation rule that captures the concept of retrieval precision for S:

$$prec_{PDA}(S, R) = \frac{1}{|R|} \sum_{r \in R} \frac{|r \sqcap \bigcup_{s \in S} s|}{|r|}, \quad (2)$$

where $\sqcap$ computes the positionally overlapping characters. The domain of $prec_{PDA}$ is $[0, 1]$; in particular it can be shown that this definition quantifies the necessary properties of a precision statistic.

Problem 2. Both the micro-averaged view and the macro-averaged view are insensitive to the number of times an $s \in S$ is detected in a detection result R, i.e., the granularity of R. We define the granularity of R for a set of plagiarized sections S by the average size of the existing covers: a detection $r \in R$ belongs to the cover C_s of an $s \in S$ iff s and r overlap. Let $S_R \subseteq S$ denote the set of cases so that for each $s \in S : |C_s| > 0$. The granularity of R given S is defined as follows:

$$gran_{PDA}(S, R) = \frac{1}{|S_R|} \sum_{s \in S_R} |C_s|, \quad (3)$$

where $S_R = \{s \mid s \in S \wedge \exists r \in R : s \cap r \neq \emptyset\}$ and $C_s = \{r \mid r \in R \wedge s \cap r \neq \emptyset\}$. The domain of the granularity is $[1, |R|]$, where 1 marks the desireable one-to-one correspondence between R and S, and where $|R|$ marks the worst case, when a single $s \in S$ is detected over an over again.

The measures (1), (2), and (3) are combined to an overall score:

$$overall_{PDA}(S, R) = \frac{F}{\log_2(1 + gran_{PDA})},$$

where F denotes the F-Measure, i.e., the harmonic mean of the precision $prec_{PDA}$ and the recall rec_{PDA}. To smooth the influence of the granularity on the overall score we take its logarithm.

4 Survey of Detection Approaches

For the competition, 13 participants developed plagiarism detection systems to tackle one or both of the tasks external plagiarism detection and intrinsic plagiarism detection. The questions that naturally arise: how do they work and how well? To give an answer, we survey the approaches in a unified way and report on their detection quality in the competition.

4.1 External Plagiarism Detection

Most of the participants competed in the external plagiarism detection task of the competition; detection results were submitted for 10 systems. As it turns out, all systems are based on common approaches—although they perform very differently.

As explained at the outset, external plagiarism detection divides into three steps (cf. Figure 1): the heuristic retrieval step, the detailed analysis step, and the post-processing step. Table 1 summarizes the participants' detection approaches in terms of these steps. However, the post-processing step was omitted here since neither of the participants applied noteworthy post-processing. Each row of the table summarizes one system; we restrict the survey to the top 6 systems since

[2] In (Stein, 2007) this idea is mathematically derived as "precision stress" and "recall stress".

Table 1: Unified summary of the detection approaches of the participants.

External Plagiarism Detection Approach		
Heuristic Retrieval	**Detailed Analysis**	**Participant**
Retrieval Model. Character-16-gram VSM (frequency weights, cosine similarity) *Comparison of D_q and D.* Exhaustive *Candidates $D_x \subset D$ for a d_q.* The 51 documents most similar to d_q.	*Exact Matches of d_q and $d_x \in D_x$.* Character-16-grams *Match Merging Heuristic to get (s_q, s_x).* Computation of the distances of adjacent matches. Joining of the matches based on a Monte Carlo optimization. Refinement of the obtained section pairs, e.g., by discarding too small sections.	Grozea, Gehl, and Popescu (2009)
Retrieval Model. Word-5-gram VSM (boolean weights, Jaccard similarity) *Comparison of D_q and D.* Exhaustive *Candidates $D_x \subset D$ for a d_q.* Documents which share at least 20 n-grams with d_q.	*Exact Matches of d_q and $d_x \in D_x$.* Word-5-grams *Match Merging Heuristic to get (s_q, s_x).* Extraction of the pairs of sections (s_q, s_x) of maximal size which share at least 20 matches, including the first and the last n-gram of s_q and s_x, and for which 2 adjacent matches are at most 49 not-matching n-grams apart.	Kasprzak, Brandejs, and Křipač (2009)
Retrieval Model. Word-8-gram VSM (frequency weights, custom distance) *Comparison of D_q and D.* Exhaustive *Candidates $D_x \subset D$ for a d_q.* The 10 documents nearest to d_q.	*Exact Matches of d_q and $d_x \in D_x$.* Word-8-grams *Match Merging Heuristic to get (s_q, s_x).* Extraction of the pairs of sections (s_q, s_x) which are obtained by greedily joining consecutive matches if their distance is not too high.	Basile et al. (2009)
Using the commercial system Plagiarism Detector (`http://plagiarism-detector.com`)		Palkovskii, Belov, and Muzika (2009)
Retrieval Model. Word-1-gram VSM (frequency weights, cosine similarity) *Comparison of D_q and D.* Clustering-based data-partitioning of D's sentences. Comparison of D_q's sentences with each partitions' centroid. *Candidates $D_x \subset D$ for a d_q.* For each sentence of d_q, the documents from the 2 most similar partitions which share similar sentences.	*Exact Matches of d_q and $d_x \in D_x$.* Sentences *Match Merging Heuristic to get (s_q, s_x).* Extraction of the pairs of sections (s_q, s_x) which are obtained by greedily joining consecutive sentences. Gaps are allowed if the respective sentences are similar to the corresponding sentences in the other document.	Muhr et al. (2009)
Retrieval Model. Winnowing fingerprinting 50 char chunks with 30 char overlap *Comparison of D_q and D.* Exhaustive *Candidates $D_x \subset D$ for a d_q.* Documents whose fingerprints share at least one value with d_q's fingerprint.	*Exact Matches of d_q and $d_x \in D_x$.* Fingerprint chunks *Match Merging Heuristic to get (s_q, s_x).* Extraction of the pairs of sections (s_q, s_x) which are obtained by enlarging matches and joining adjacent matches. Gaps must be below a certain Levenshtein distance.	Scherbinin and Butakov (2009)

the overall performance of the remaining systems is negligible. Nevertheless, these systems also implement the generic three-step process. The focus of this survey is on describing algorithmic and retrieval aspects rather than implementation details. The latter are diverse in terms of applied languages, software, and their runtime efficiency; descriptions can be found in the respective references.

The heuristic retrieval step (column 1 of Table 1) involves the comparison of the corpus' suspicious documents D_q with the source documents D. For this, each participant em-

Table 2: Performance results for the external plagiarism detection task.

		External Detection Quality				
Rank	**Overall**	**F**	**Precision**	**Recall**	**Granularity**	**Participant**
1	0.6957	0.6976	0.7418	0.6585	1.0038	Grozea, Gehl, and Popescu (2009)
2	0.6093	0.6192	0.5573	0.6967	1.0228	Kasprzak, Brandejs, and Křipač (2009)
3	0.6041	0.6491	0.6727	0.6272	1.1060	Basile et al. (2009)
4	0.3045	0.5286	0.6689	0.4370	2.3317	Palkovskii, Belov, and Muzika (2009)
5	0.1885	0.4603	0.6051	0.3714	4.4354	Muhr et al. (2009)
6	0.1422	0.6190	0.7473	0.5284	19.4327	Scherbinin and Butakov (2009)
7	0.0649	0.1736	0.6552	0.1001	5.3966	Pereira, Moreira, and Galante (2009)
8	0.0264	0.0265	0.0136	0.4586	1.0068	Vallés Balaguer (2009)
9	0.0187	0.0553	0.0290	0.6048	6.7780	Malcolm and Lane (2009)
10	0.0117	0.0226	0.3684	0.0116	2.8256	Allen (2009)

ploys a specific retrieval model, a comparison strategy, and a heuristic to select the candidate documents D_x from the D. Most of the participants use a variation of the well-known vector space model (VSM) as retrieval model, whereas, the tokens are often character- or word-n-grams instead of single words. As comparison strategy, the top 3 approaches perform an exhaustive comparison of D_q and D, i.e., each $d_q \in D_q$ is compared with each $d_x \in D$ in time $O(|D_q| \cdot |D|)$, while the remaining approaches employ data partitioning and space partitioning technologies to achieve lower runtime complexities. To select the candidate documents D_x for a d_q either its k nearest neighbors are selected or the documents which exceed a certain similarity threshold.

The detailed analysis step (column 2 of Table 1) involves the comparison of each $d_q \in D_q$ with its respective candidate documents D_x in order to extract pairs of sections (s_q, s_x), where $s_q \in d_q$ and $s_x \in d_x$, $d_x \in D_x$, from them which are highly similar, if any. For this, each participant first extracts all exact matches between d_q and d_x and then merges the matches heuristically to form suspicious sections (s_q, s_x). While each participant uses the same type of token to extract exact matches as his respective retrieval model of the heuristic retrieval step, the match merging heuristics differ largely from one another. However, it can be said that in most approaches a kind of distance between exact matches is measured first, and then a custom algorithm is employed which clusters them to sections.

Table 2 lists the detection performance results of all approaches, computed with the quality measures introduced in Section 3. Observe that the approach with top precision is the one on rank 6 which is based on fingerprinting, the approach with top recall is the one on rank 2, and the approach with top granularity is the one on rank 1. The latter is also the winner of this task since it provides the best trade off between the three quality measures.

4.2 Intrinsic Plagiarism Detection

The intrinsic plagiarism detection task has gathered less attention than external plagiarism detection; detection results were submitted for 4 systems. Table 3 lists their detection performance results. Unlike in external plagiarism detection, in this task the baseline performance is not 0. The reason for this is that intrinsic plagiarism detection is a one-

Table 3: Performance results for the intrinsic plagiarism detection task.

		Intrinsic Detection Quality					
Rank	**Overall**	**F**	**Precision**	**Recall**	**Granularity**	**Participant**	
1	0.2462	0.3086	0.2321	0.4607	1.3839	Stamatatos (2009)	
2	0.1955	0.1956	0.1091	0.9437	1.0007	Hagbi and Koppel (2009)	(Baseline)
3	0.1766	0.2286	0.1968	0.2724	1.4524	Muhr et al. (2009)	
4	0.1219	0.1750	0.1036	0.5630	1.7049	Seaward and Matwin (2009)	

Table 4: Overall plagiarism detection performance.

		Overall Detection Quality				
Rank	**Overall**	**F**	**Precision**	**Recall**	**Granularity**	**Participant**
1	0.4871	0.4884	0.5193	0.4610	1.0038	Grozea, Gehl, and Popescu (2009)
2	0.4265	0.4335	0.3901	0.4877	1.0228	Kasprzak, Brandejs, and Křipač (2009)
3	0.4229	0.4544	0.4709	0.4390	1.1060	Basile et al. (2009)
4	0.2131	0.3700	0.4682	0.3059	2.3317	Palkovskii, Belov, and Muzika (2009)
5	0.1833	0.4001	0.4826	0.3417	3.5405	Muhr et al. (2009)
6	0.0996	0.4333	0.5231	0.3699	19.4327	Scherbinin and Butakov (2009)
7	0.0739	0.0926	0.0696	0.1382	1.3839	Stamatatos (2009)
8	0.0586	0.0587	0.0327	0.2831	1.0007	Hagbi and Koppel (2009)
9	0.0454	0.1216	0.4586	0.0701	5.3966	Pereira, Moreira, and Galante (2009)
10	0.0366	0.0525	0.0311	0.1689	1.7049	Seaward and Matwin (2009)
11	0.0184	0.0185	0.0095	0.3210	1.0068	Vallés Balaguer (2009)
12	0.0131	0.0387	0.0203	0.4234	6.7780	Malcolm and Lane (2009)
13	0.0081	0.0157	0.2579	0.0081	2.8256	Allen (2009)

class classification problem in which it has to be decided for each section of a document whether it is plagiarized, or not. The baseline performance in such problems is commonly computed as the naive assumption that everything belongs to the target class, which is also what Hagbi and Koppel (2009) did who classified almost everything as plagiarized. Interestingly, the baseline approach is on rank 2 while two approaches perform worse than the baseline. Only the approach of Stamatatos (2009) performs better than the baseline.

4.3 Overall Detection Results

To determine the overall winner of the competition, we have computed the combined detection performance of each participant on the competition corpora of both tasks. Table 4 shows the results. Note that the competition corpus of the external plagiarism detection task is a lot bigger than the one for the intrinsic plagiarism detection task, which is why the top ranked approaches are those who performed best in the former task. Overall winner of the competition is the approach of Grozea, Gehl, and Popescu (2009).

5 Summary

The 1st International Competition on Plagiarism Detection fostered research and brought a number of new insights into the problems of automatic plagiarism detection and its evaluation. An important by-product of the competition is a controlled large-scale evaluation framework which consists of a corpus of artificial plagiarism cases and new detection quality measures. The corpus contains more than 40 000 documents and about 94 000 cases of plagiarism.

Furthermore, in this paper we give a comprehensive overview about the competition and in particular about the plagiarism detection approaches of the competition's 13 participants. It turns out that all of the detection approaches follow a generic retrieval process scheme which consists of the three steps heuristic retrieval, detailed analysis, and knowledge-based post-processing. To ascertain this fact we have compiled a unified summary of the top approaches in Table 1.

The competition divided into the two tasks external plagiarism detection and intrinsic plagiarism detection. The winning approach for the former task achieves 0.74 precision at 0.65 recall at 1.00 granularity. The winning approach for the latter task improves 26% above the baseline approach and achieves 0.23 precision at 0.46 recall at 1.38 granularity.

Acknowledgements

We thank Yahoo! Research and the University of the Basque Country for their sponsorship. This work was also partially funded by the Text-Enterprise 2.0 TIN2009-13391-C04-03 project and the CONACYT-MEXICO 192021 grant. Our special thanks go to the participants of the competition for their devoted work.

References

Allen, James. 2009. Submission to the 1st International Competition on Plagiarism Detection. From the Southern Methodist University in Dallas, USA.

Basile, Chiara, Dario Benedetto, Emanuele Caglioti, Giampaolo Cristadoro, and Mirko Degli Esposti. 2009. A Plagiarism Detection Procedure in Three Steps: Selection, Matches and "Squares". In Stein et al. (Stein et al., 2009).

Clough, Paul. 2003. Old and new challenges in automatic plagiarism detection. National UK Plagiarism Advisory Service, http://ir.shef.ac.uk/cloughie/papers/pas_plagiarism.pdf.

Grozea, Cristian, Christian Gehl, and Marius Popescu. 2009. ENCOPLOT: Pairwise Sequence Matching in Linear Time Applied to Plagiarism Detection. In Stein et al. (Stein et al., 2009).

Hagbi, Barak and Moshe Koppel. 2009. Submission to the 1st International Competition on Plagiarism Detection. From the Bar Ilan University, Israel.

Kasprzak, Jan, Michal Brandejs, and Miroslav Křipač. 2009. Finding Plagiarism by Evaluating Document Similarities. In Stein et al. (Stein et al., 2009).

Malcolm, James A. and Peter C. R. Lane. 2009. Tackling the PAN'09 External Plagiarism Detection Corpus with a Desktop Plagiarism Detector. In Stein et al. (Stein et al., 2009).

Maurer, Hermann, Frank Kappe, and Bilal Zaka. 2006. Plagiarism - a survey. *Journal of Universal Computer Science*, 12(8):1050–1084.

Meyer zu Eissen, Sven and Benno Stein. 2006. Intrinsic plagiarism detection. In Mounia Lalmas, Andy MacFarlane, Stefan M. Rüger, Anastasios Tombros, Theodora Tsikrika, and Alexei Yavlinsky, editors, *Proceedings of the European Conference on Information Retrieval (ECIR 2006)*, volume 3936 of *Lecture Notes in Computer Science*, pages 565–569. Springer.

Muhr, Markus, Mario Zechner, Roman Kern, and Michael Granitzer. 2009. External and Intrinsic Plagiarism Detection Using Vector Space Models. In Stein et al. (Stein et al., 2009).

Palkovskii, Yurii Anatol'yevich, Alexei Vitalievich Belov, and Irina Alexandrovna Muzika. 2009. Submission to the 1st International Competition on Plagiarism Detection. From the Zhytomyr State University, Ukraine.

Pereira, Rafael C., V. P. Moreira, and R. Galante. 2009. Submission to the 1st International Competition on Plagiarism Detection. From the Universidade Federal do Rio Grande do Sul, Brazil.

Scherbinin, Vladislav and Sergey Butakov. 2009. Using Microsoft SQL Server Platform for Plagiarism Detection. In Stein et al. (Stein et al., 2009).

Seaward, Leanne and Stan Matwin. 2009. Intrinsic Plagiarism Detection Using Complexity Analysis. In Stein et al. (Stein et al., 2009).

Stamatatos, Efstathios. 2009. Intrinsic Plagiarism Detection Using Character n-gram Profiles. In Stein et al. (Stein et al., 2009).

Stein, Benno. 2007. Principles of hash-based text retrieval. In Charles Clarke, Norbert Fuhr, Noriko Kando, Wessel Kraaij, and Arjen de Vries, editors, *30th Annual International ACM SIGIR Conference*, pages 527–534. ACM, July.

Stein, Benno, Sven Meyer zu Eissen, and Martin Potthast. 2007. Strategies for Retrieving Plagiarized Documents. In Charles Clarke, Norbert Fuhr, Noriko Kando, Wessel Kraaij, and Arjen de Vries, editors, *30th Annual International ACM SIGIR Conference*, pages 825–826. ACM, July.

Stein, Benno, Paolo Rosso, Efstathios Stamatatos, Moshe Koppel, and Eneko Agirre, editors. 2009. *Proceedings of the SEPLN Workshop on Uncovering Plagiarism, Authorship, and Social Software Misuse, PAN'09, September 10 2009, Donostia-San Sebastián, Spain.* Universidad Polytécnica de Valencia.

Vallés Balaguer, Enrique. 2009. Putting Ourselves in SME's Shoes: Automatic Detection of Plagiarism by the WCopyFind tool. In Stein et al. (Stein et al., 2009).

Weber-Wulff, Debora and Katrin Köhler. 2008. Plagiarism detection softwaretest 2008. http://plagiat.htw-berlin.de/software/2008/.

Webis at Bauhaus-Universität Weimar and NLEL at Universidad Politécnica de Valencia. 2009. PAN Plagiarism Corpus PAN-PC-09. http://www.webis.de/research/corpora. Martin Potthast, Andreas Eiselt, Benno Stein, Alberto Barrón-Cedeño, and Paolo Rosso (editors).

ENCOPLOT: Pairwise Sequence Matching in Linear Time Applied to Plagiarism Detection *

Cristian Grozea
Fraunhofer FIRST
IDA Group
Kekulestrasse 7,
12489 Berlin, Germany
cristian.grozea@first.fraunhofer.de

Christian Gehl
Fraunhofer FIRST
IDA Group
Kekulestrasse 7,
12489 Berlin, Germany
christian.gehl@first.fraunhofer.de

Marius Popescu
University of Bucharest
Faculty of Mathematics and Computer Science
Academiei 14, Sect. 1,
Bucharest, Romania
popescunmarius@gmail.com

Abstract: In this paper we describe a new general plagiarism detection method, that we used in our winning entry to the 1^{st} International Competition on Plagiarism Detection, the external plagiarism detection task, which assumes the source documents are available. In the first phase of our method, a matrix of kernel values is computed, which gives a similarity value based on n-grams between each source and each suspicious document. In the second phase, each promising pair is further investigated, in order to extract the precise positions and lengths of the subtexts that have been copied and maybe obfuscated – using *encoplot*, a novel linear time pairwise sequence matching technique. We solved the significant computational challenges arising from having to compare millions of document pairs by using a library developed by our group mainly for use in network security tools. The performance achieved is comparing more than 49 million pairs of documents in 12 hours on a single computer. The results in the challenge were very good, we outperformed all other methods.

Keywords: n-gram, plagiarism detection, network security, challenge

1 Introduction

Many methods have been developed for plagiarism detection, especially for the *external plagiarism analysis*, which consists in finding passages in the suspicious documents which have been plagiarized and the corresponding text passages in the source documents. Almost all these methods handle the text at word level. Various comparison units have been employed in plagiarism detection methods. Entire documents are compared in (Lyon, Barrett, and Malcolm, 2004). Sentences from suspicious documents are compared to sentences from reference documents in (Kang, Gelbukh, and Han, 2006). Mixed-length comparisons in which suspicious sentences are compared with entire reference documents were used in (Barrón-Cedeño and Rosso, 2009; Barrón-Cedeño, Rosso, and Benedí, 2009). Irrespective of the comparison unit used, all methods of plagiarism detection need a similarity measure to compare the text fragments corresponding to the comparison unit. Most similarity measures used in plagiarism detection are based on estimating the amount of common configurations of words. They differ by the configurations considered (n-grams, subsequences, etc.) or by what words are used in comparisons (only words from the text fragments, stemmed or not, synonyms from WordNet, etc.). In (Lyon, Barrett, and Malcolm, 2004) word trigrams are used to measure the similarity between texts. The authors based their choice of using word trigrams for plagiarism detection on the fact that the number of common word trigrams in two independently written texts (even if the text are on the same topic) must be low given the Zipfian distribution of words. Also in (Barrón-Cedeño and Rosso, 2009) it is reported that using word bigrams and trigrams led to best results in their experiments. In order to address the prob-

* partly supported from the BMBF project ReMIND (KZ 01-IS07007A) and CNCSIS PN2-Idei project 228

lem of *rewording* in plagiarism, **PPChecker** (Kang, Gelbukh, and Han, 2006) is based on a special designed similarity measure, that takes into account also the synonyms (obtained from the WordNet) of the words in the suspicious sentences. Some of the most elaborate similarity measures used in plagiarism detection are described in (Bao et al., 2003; Bao et al., 2004a; Bao et al., 2004b). These measures are derived from the *string kernel*, a kernel type successfully used in text categorization (Lodhi et al., 2002). The string kernel works at character level, although in (Bao et al., 2003; Bao et al., 2004a; Bao et al., 2004b) it is extended to work at word level, comparing two semantic sequences according to their common words and position information.

Using words is natural in text analysis tasks like text categorization (by topic), authorship identification and plagiarism detection. Perharps surprisingly, recent results proved that methods that handle the text at character level can also be very effective in text analysis tasks. In (Lodhi et al., 2002) string kernels were used for document categorization with very good results. Trying to explain why treating documents as symbol sequences and using string kernels obtained such good results the authors suppose that: "the [string] kernel is performing something similar to stemming, hence providing semantic links between words that the word kernel must view as distinct". String kernels were also successfully used in authorship identification (Sanderson and Guenter, 2006; Popescu and Dinu, 2007). A possible reason for the success of string kernels in authorship identification is given in (Popescu and Dinu, 2007): "the similarity of two strings as it is measured by string kernels reflects the similarity of the two texts as it is given by the short words (2-5 characters) which usually are function words, but also takes into account other morphemes like suffixes ('ing' for example) which also can be good indicators of the author's style"[1]

For plagiarism detection, the only approach that handles the text at character level that we are aware of is in (Bao, Lyon, and Lane, 2006), for Chinese, and there is justified by the difficulties of the Chinese language (word segmentation).

There is a strong connection between the research in NLP and the research in computer network security. In recent years, network security research started to approach the problem of detecting automatically unknown attacks as soon as they reach the targeted system. These attacks may follow the syntax but try to exploit the semantics of the network communication between the client and the server applications, in order to gain access over the attacked computer or at least to prevent it from working normally. The communication process defined by the application layer protocols – e.g. HTTP, FTP, RPC or IMAP – can also be considered as a text-based communication in an artificial language. The idea of payload analysis, which treats the data as sequences of bytes has been explored in detail (Kruegel, Toth, and Kirda, 2002; Wang and Stolfo, 2004; Rieck and Laskov, 2006; Wang, Parekh, and Stolfo, 2006; Rieck and Laskov, 2007). As the focus in this field shifted towards applying more advanced machine learning methods, generalizing the extraction and representation of the features has increased much the flexibility in defining similarity measures between sequential data, in a security context. The work (Rieck and Laskov, 2008) presents an efficient way to combine features extracted from byte sequences, e.g. *words* or n-grams with arbitrary n value, for a wide range of linear and non-linear similarity measures.

Graphics methods in comparing sequences have been used in many fields, mostly under the name *dotplot* – see (Maizel and Lenk, 1981) for one of the first uses in biology and (Church and Helfman, 1993) for uses in source text comparison. Whereas very attractive for exploratory data analysis, building this graphic is potentially quadratic in time and space. Also it tends to be noisy, by showing many irrelevant coincidences between the sequences compared. Even with these limitations, the method has been applied to source code, videos, music, protein and other biological sequences, with various ways to filter the noisy graphics and to handle the problem of the potential quadratic size. We improve on this technique by deriving our own, linear space, linear time technique, that we named the *encoplot*, short for "eN-gram COincidence PLOT". It is fully described in Section 2.3, with code in Appendix 1.

[1] the string kernel used in (Popescu and Dinu, 2007) takes into account substrings of length up to 5 characters.

Our plagiarism detection method can be described as a combination of techniques from many fields: it is character n-gram based. It leverages a very efficient network security software to compute the matrices of kernel values. It uses the very fast *encoplot* algorithm and processes the *encoplot* data in a quantitative fashion to solve what can be seen as a rudimentary machine vision or a specialized 2-dimensional data clustering task, in order to identify the matching text passages for a given document pair, as explained thoroughly below.

In what follows, the dataset specifics and the time performance figures refer to the dataset of the 1^{st} International Competition on Plagiarism Detection, external plagiarism detection task (Webis at Bauhaus-Universität Weimar and NLEL at Universidad Politécnica de Valencia, 2009). The development corpus of this dataset contained about 7000 source documents and 7000 suspicious ones, with the plagiarism generated automatically with various degrees of obfuscation (permutations, words deleted, inserted or replaced by synonyms or antonyms) and annotated. The competition corpus had the same characteristics (different documents) and the annotation was missing.

2 Methods

Our approach consists of two main phases. In the first phase, a matrix of string kernel values is computed, which gives a similarity value between each source and each suspicious document. Then, for each source, the possible "destinations" (suspicious documents) are ranked based on their similarity level with the current source, in decreasing order. In the second phase, each promising pair is further investigated, in order to extract the precise positions and lengths of the subtexts that have been copied and maybe obfuscated by the random plagiarist. In the end we do a supplementary filtering that increases the precision with the price of decreasing the recall.

2.1 Selecting a kernel and computing the matrix of kernel values for a large set of documents

Based on the work of (Rieck and Laskov, 2008), a *C* library for sequential data, *libmindy*, has been implemented by our network security research group. It has been developed mainly for being used in building real-time network analysis tools at packet level, as part of network intrusion detection and prevention systems. It can map byte sequences to a vectorial n-gram representation, such that the similarity between two byte sequences can be expressed in terms of distance and kernel functions on those representations. The n-gram extraction set of feasible byte sequences is given by $A_n = \Sigma^n$, where Σ is the alphabet (in our case the whole ASCII–8 set). The n-gram embedding function ϕ for a byte sequence x is then defined as $\phi(x) = (\phi_{ng}(x))_{ng \in A_n}$ with $\phi_{ng}(x) = \mathrm{emb}(x, ng)$, where the dimension of the vector $\phi(x)$ is $|A_n|$. The function $\mathrm{emb}(x, ng)$ returns either the frequency, the count or the presence bit for a n-gram ng in x. With the embedding function ϕ fixed, one can compute a pairwise similarity value for the vectorial representations of two byte sequences. Table 1 presents a selection of the implemented distances and similarity measures that we could have used (where x and y are arbitrary byte sequences).

distance function $\mathbf{d(x, y)}$	
Minkowski	$\sqrt[k]{\sum_{ng \in A_n} \lvert\phi_{ng}(x) - \phi_{ng}(y)\rvert^k}$
Canberra	$\sum_{ng \in A_n} \frac{\lvert\phi_{ng}(x) - \phi_{ng}(y)\rvert}{\phi_{ng}(x) + \phi_{ng}(y)}$
kernel function $\mathbf{k(x, y)}$	
linear kernel	$\sum_{ng \in A_n} \phi_{ng}(x) \cdot \phi_{ng}(y)$
RBF kernel	$exp(-\frac{\sum_{ng \in A_n} \lvert\lvert\phi_{ng}(x) - \phi_{ng}(y)\rvert\rvert^2}{2\sigma^2})$

Table 1: Distances and kernels functions for sequential data.

Experiments with a very small subset of only 5 documents and our previous experience in string kernels led us to use the linear kernel over a representation where every n-gram present is marked by 1 and every other is marked by 0 (ignoring thus the frequencies of the n-grams). The kernel was normalized, such as $K(x, x) = 1$ for any string x. For the length of the n-grams we used 16 characters. Although in our estimations 18 should have been better (closer to three times the average word length plus two separators), the speed-up of the software used can only be obtained up to n-grams of length 16, see below and Appendix 1 for details. Using windows of two to three words in plagiarism detection was

found to be the best choice by (Lyon, Barrett, and Malcolm, 2004) and (Barrón-Cedeño and Rosso, 2009).

The computation of a matrix of kernel values with sizes as large as 7000 is computationally intensive. There are more than 49 million pairs of documents for which the kernel value has to be computed, in each of the two datasets, the development and the competition corpus, accounting for a total of more than 98 million pairs to consider. *libmindy* has had already a tool for building a (symmetric) kernel matrix for a set of documents. We extended this tool for being able to handle asymmetric matrices of kernel values, where the kernel values are computed for each $x \in X$ and $y \in Y$, where X and Y are two independent finite sets of files, not necessarily having the same cardinal. While the new tool could in principle perform the task fast enough, it would have needed an amount of RAM of about 400 GB for a kernel based on length 16 n-grams. To avoid this issue, we partitioned the matrix of kernel values in blocks of sizes up to 1000x1000 (1 million pairs in most blocks), which required only 8 to 10 GB of RAM for processing. Those 64 blocks per dataset we processed one after the other, but the processing of each block was fully parallelized on the 8 cores of the machine, as a result of internally distributing the tasks by the means of OpenMP programming. Processing a full dataset took 12 hours on the machine we used (Dell Precision T7400). Although we had access to a cluster, it offered only a 32-bit environment. This would have slowed the whole processing by a factor that would almost completely eliminated the advantage of having 8 to 12 times more computing cores, and this is why we decided to use a single multi-core computer.

2.2 Pruning of the pairs

If the total processing for one pair of documents (up to book length level) would only take one second, this would lead to a total computation time of more than three years! Even by successfully parallelizing this task and dividing the time by hopefully 8 (the number of computing cores), the time needed would have been more than 4 months. It was obvious that even with the matrix of kernel values computed, there is too much work in comparing the documents in each pair. Pruning was seen from the start as a requirement, the question was what effect will it have on limiting the performance that can be achieved. We have considered ranking the pairs such that the ones with most chances of corresponding to plagiarism come first. Ranking on the absolute values of the kernel proved to work worst. Ranking for each source the suspicious documents proved to provide a consistent 10% advantage over ranking for each suspicious document the sources. Therefore, given also the values that can be seen in the Figure 1, we decided to limit our effort to the first 51 most promising suspicious documents for each given source.

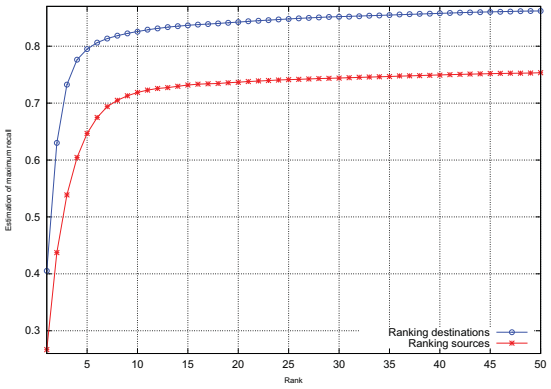

Figure 1: Maximum achievable recall for different pruning thresholds. Ranking the suspicious documents for each source leads consistently to better values than ranking the sources for each suspicious document.

2.3 Comparing two documents - The *encoplot*

With the maximum effort down to an estimate of about 100 hours, assuming spending in average a second per exhaustive document comparison (with the hope of reducing it to 12 hours by multicore parallelism), we proceeded to search for a way to identify what the documents have in common, if anything. Essential to this was the visualization of the coincidence pattern of n-grams between two documents. This is a scatter plot of a sublist of the positions where both texts have the same n-gram. We call this plot *encoplot*. Plots computed for pairs in the development corpus can be seen in Figures 2 and 3. All these plots use documents in the development dataset.

Related ideas (the “dotplot” graphs) exist about visualizing the n-grams that two texts (or sequences) share. The problem with those

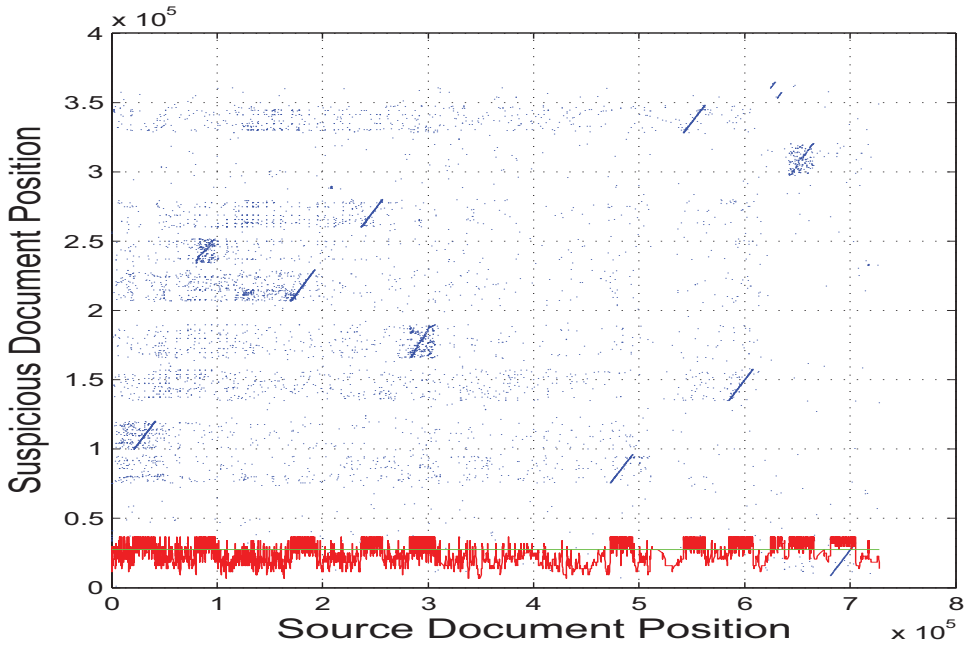

Figure 2: Encoplot for source #3094 and suspicious #9. Many plagiarism instances for the same document pair. The shattered look of some comes from higher obfuscation. In red, the local contiguity score, scaled.

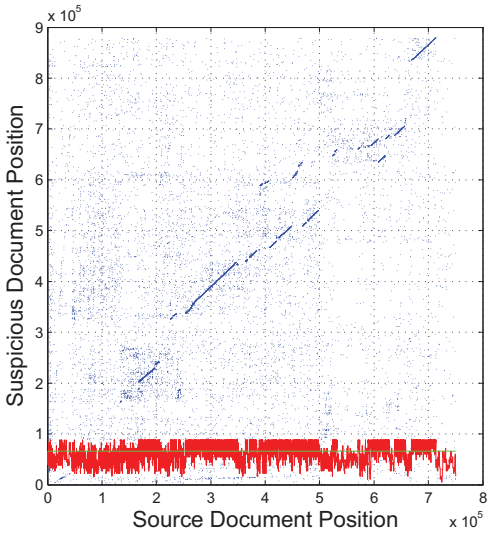

Figure 3: Encoplot for source #134 and suspicious #2499 – a real case of human (self) plagiarism.

is that the number of pairs can be quadratic in the size of the documents. For megabytes long texts, this can easily become computationally intractable. We solve this issue by limiting ourselves to a sublist that is guaranteed to be no longer than the shortest of the documents, and can be computed in linear time. The precise procedure we employed starts by sorting virtually the sets of n-grams for both documents to be compared. Then these ordered sets of n-grams are compared with a procedure that is derived from the procedure from merging two sorted lists. Every time the smallest elements of the two lists differ, the smallest of them is dropped, without producing any output. Every time the smallest elements of the lists are equal, the pair of positions on which this identical n-gram occurs is being collected by outputting it to the standard output. Code for this core procedure is given in Appendix 1. Please note that *encoplot* pairs the first instance of an n-gram in one document with the first instance of the same in the other document, the second one with the second one and so on – as opposed to the dotplot, wich pairs each instance with each instance.

2.4 Heuristics used for separating the copied subtexts

Once the *encoplot* data (the list of pairs of indexes) is obtained, it is sorted by the value of the first index in each pair, which corresponds to the position in source of the common n-gram. From this list a local "contiguity" score is derived by computing whether there is simultaneously a small jump on both indexes (sum of absolute jumps less than 4) when going from a pair to the next pair, followed by a smoothing by a convolution with a constant vector of length 16. The contigu-

ity score for an *encoplot* is displayed in red in Figures 2 and 2. Then a Monte Carlo optimization procedure is called, not more than 30 times for each document pair, which in 10 attempts tries to find the largest group from the current *encoplot* data. The start of the group is decided randomly with uniform distribution over the list of available pairs, then the group is extended to left and right such that the average contiguity score stays above 0.5 and there are no jumps (skipped portions) longer than 512 in any 16 steps. After a group is obtained, it is checked to have an average contiguity score of over 0.75 and a length of at least 256 characters. If not, it is rejected as insignificant. If kept, it is projected to the dimension of the indexes that correspond to the suspicious document, and only the compact core of it is preserved. The compact core is obtained by sorting on the suspicious document axis and eliminating the outliers by starting from the group center and extending it to left and right while the skips are less than 256 positions. What remains is projected back onto the source document axis, obtaining thus an estimate of the indexes whose convex hull define the two subtexts corresponding to each other. This candidate of a plagiarism instance is checked once again, this time for a final length of at least 256, for not having shrinked to less than half with respect to the initial group length and for the two subtexts not having sizes too different (the absolute difference more than half of the mean of the two lengths). This subset of the *encoplot* data is removed, the plagiarism instance is outputted if all tests succeeded, and the procedure is repeated in the search for more groups. If the group found fails to satisfy the checks, it is deemed as a failure. At three consecutive failures the search is abandoned and the treatment of the pair of documents is considered completed. This decision may be risky, but accelerates substantially this phase, as on very complicated document pairs it can take minutes to completely examine an involved pair. On the other hand, for the actually unrelated documents this ends the investigation rapidly. Technically, we have accelerated this processing phase even more by running simultaneously up to 10 detailed examinations of document pairs at a time, trying to balance the processing power required and the disk latency.

3 Results

We combined the best *F-measure* – the harmonic mean of precision and recall – 0.6976 (the next competitor had 0.6192) with the best granularity – lack of fragmentation in detection of the plagiated passages – 1.0027 (the next best value was 1.0164), winning thus the competition.

4 Discussion and Conclusions

The first question is whether our choice to compare the documents in pairs was optimal. Indexing based methods could be faster, by eliminating the need for exhaustive pairwise comparison of documents in a large corpus. They function by first indexing the collection of source documents and then searching for parts of the suspicious documents in the index, as the system MOSS (Schleimer, Wilkerson, and Aiken, 2003) does. Such an inflexible approach cannot handle well obfuscation, as opposed to our approach. On the other hand, flexible matching is an always-current research topic in information retrieval systems (Navarro, 2001), and this eventually improves plagiarism detection as well. We think that, whereas needing more computational effort, our approach had the chance of producing better results. And, as a consequence of using highly optimized network analysis code, it did so in a reasonable time, even when run on a single contemporary computer, as opposed to a full cluster. One could say that it was closer to being optimal in terms of quality of the results, while still being acceptable in terms of running time.

A second question of interest is whether our values for the hyperparameters of the method are optimal for this dataset. The answer is probably no, but maybe not far from that. They have been chosen by educated guess guided by the exploratory data analysis, as opposed to blindly optimizing a cross-validation towards the best (over)fitting.

The third interesting issue is the claim of some experts that only the humans can have very good results at spotting plagiarism (Weber-Wulff, 2008). We think that, as far as the ethics is concerned, a human must look at the evidence before claiming a case as one of plagiarism. And of course, text understanding is still not within the reach of artificial intelligence yet. On the other hand, the claim that the only automatization in plagiarism detection should limit to using

the one's favorite search engine and searching for paragraphs selected based on one's intuition is questionable. How would such an expert deal with 7000 documents up to a book length? How long would it take to process those by hand, even using a public search engine? How long does it take one to read 7000 works/books? The need for automatization seems evident, as it was to (Grozea, 2004) when he had to grade 400 projects from 60 students in less than 24 hours. Crowdsourcing could also be a possibility, but one needs very big crowds for that (optimally quadratic size, if using the same choice in the trade-off between speed and quality as we chose). Time is the key factor in plagiarism detection.

Given the very good results obtained by our method it is worth asking – and further investigating – whether using character n-grams offers any advantage over using word n-grams. First, let us note that our method uses n-grams of 16 characters which in average[2] correspond to word trigrams (the standard approach in plagiarism detection). It may seem that (on average) the same information is brought by 16 characters n-grams and word trigrams. What differentiates the two types of n-grams is in our opinion the fact that character n-grams favor long words over short ones, and when people copy text they do that for the content words of the copied text that tend to be longer than the functional words (stop words) which are short. For example: a common syntagmatic expression[3] like "as far as" will contribute with one word trigram, but with none character 16-gram. On the other hand, a sequence of content words (worth being copied) like "educated guess guided" will contribute again with only one word trigram, but with 6 character 16-grams.

Another item to discuss is how to balance precision and recall in automatic plagiarism detection systems. Given that a human is in many cases the final link in the chain that leads to the proof of plagiarism, the effort of that human must be spared as much as possible. The accuse of plagiarism is so strong, that it needs strong evidence. Both these aspects recommend to balance the precision and recall towards a high precision, even at the expense of lowering the recall. This is how we tuned our system's parameters, including but not limited to the last checking phase. Of course, accurate comparison of systems should take into account the entire precision-recall curve. By plotting on the same graph these curves for more systems, one could easily see where is the best performance region for each system and whether or not one of the systems is overall better than another system.

Related to the maximum achievable precision while keeping a fair recall is the issue of the documents independence and of the automatic plagiarism. The dataset contains plagiarism built automatically and randomly and only these borrowings between the source documents and the suspicious documents had to be found. But the documents were not independent enough: there are pairs of documents with the same or almost the same content, such as independent translations of "One Thousand and One Night" or several Bible editions, authors doing heavy reuse from their previous works (the so-called self-plagiarism). These are interesting in two ways: they are better examples of what the human plagiarism is, so spotting those as related is very good. On the other hand, this can be seen as unintended (by the organizers) plagiarism, so any such pair reported will actually lower the precision score.

A very interesting issue is the asymmetry of the ranking quality. Why is it 10% better to rank all suspicious documents for any fixed source instead of ranking all possible sources for every fixed suspicious document, as clearly seen in Figure 1? A possible source of this asymmetry is that while it was guaranteed for each suspicious document that the areas plagiated do not overlap, this was not the case for the source documents, where the areas plagiated could overlap. This asymmetry deserves more investigation, being one of the few glints of hope so far to tackling what could be the biggest open problem in automatic plagiarism detection, that is determining the direction of plagiarism in a pair of documents – being able to indicate with confidence which is the copy and which is the original.

To conclude, by combining advanced software engineering and effort-sparing heuristics tuned using the novel visualization technique *encoplot*, we have been able to achieve the top

[2] The average word length in the corpus is 5.2

[3] Frequently and systematically co-occurring lexical items.

placement in the final results, proving that the interaction of NLP researchers with networks security researchers can lead to high-performance NLP systems.

4.1 Acknowledgment

We thank Dr. Andreas Ziehe and the anonymous reviewers for the thorough review of our paper and for the useful suggestions.

References

Bao, Jun Peng, Caroline Lyon, and Peter C. R. Lane. 2006. Copy detection in chinese documents using ferret. *Language Resources and Evaluation*, 40(3-4):357–365.

Bao, Jun-Peng, Jun-Yi Shen, Xiao-Dong Liu, Hai-Yan Liu, and Xiao-Di Zhang. 2003. Document copy detection based on kernel method. In *Proceedings of Natural Language Processing and Knowledge Engineering Conference (IEEE)*, pages 250–255.

Bao, Jun-Peng, Jun-Yi Shen, Xiao-Dong Liu, Hai-Yan Liu, and Xiao-Di Zhang. 2004a. Finding plagiarism based on common semantic sequence model. In Qing Li, Guoren Wang, and Ling Feng, editors, *WAIM*, volume 3129 of *Lecture Notes in Computer Science*, pages 640–645. Springer.

Bao, Jun-Peng, Jun-Yi Shen, Xiao-Dong Liu, Hai-Yan Liu, and Xiao-Di Zhang. 2004b. Semantic sequence kin: A method of document copy detection. In Honghua Dai, Ramakrishnan Srikant, and Chengqi Zhang, editors, *PAKDD*, volume 3056 of *Lecture Notes in Computer Science*, pages 529–538. Springer.

Barrón-Cedeño, Alberto and Paolo Rosso. 2009. On Automatic Plagiarism Detection based on n-grams Comparison. In Mohand Boughanem, Catherine Berrut, Josiane Mothe, and Chantal Soulé-Dupuy, editors, *ECIR 2009*, volume 5478 of *LNCS*, pages 696–700, Toulouse, France. Springer.

Barrón-Cedeño, Alberto, Paolo Rosso, and José-Miguel Benedí. 2009. Reducing the Plagiarism Detection Search Space on the Basis of the Kullback-Leibler Distance. In Alexander F. Gelbukh, editor, *CICLing 2009*, volume 5449 of *Lecture Notes in Computer Science*, pages 523–534, Mexico, Mexico. Springer.

Church, K.W. and J.I. Helfman. 1993. Dotplot: A program for exploring self-similarity in millions of lines of text and code. *Journal of Computational and Graphical Statistics*, pages 153–174.

Grozea, C. 2004. Plagiarism detection with state of the art compression programs. Report CDMTCS-247, Centre for Discrete Mathematics and Theoretical Computer Science, University of Auckland, Auckland, New Zealand, August.

Kang, NamOh, Alexander F. Gelbukh, and Sang-Yong Han. 2006. Ppchecker: Plagiarism pattern checker in document copy detection. In Petr Sojka, Ivan Kopecek, and Karel Pala, editors, *TSD*, volume 4188 of *Lecture Notes in Computer Science*, pages 661–667. Springer.

Kruegel, C., T. Toth, and E. Kirda. 2002. Service specific anomaly detection for network intrusion detection. In *Proc. of ACM Symposium on Applied Computing*, pages 201–208.

Lodhi, Huma, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Christopher J. C. H. Watkins. 2002. Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444.

Lyon, Caroline, Ruth Barrett, and James Malcolm. 2004. A theoretical basis to the automated detection of copying between texts, and its practical implementation in the ferret plagiarism and collusion detector. In *Plagiarism: Prevention, Practice and Policies Conference*.

Maizel, J.V. and R.P. Lenk. 1981. Enhanced graphic matrix analysis of nucleic acid and protein sequences. *Proceedings of the National Academy of Sciences*, 78(12):7665–7669.

Navarro, G. 2001. A guided tour to approximate string matching. *ACM Computing Surveys (CSUR)*, 33(1):31–88.

Popescu, Marius and Liviu P. Dinu. 2007. Kernel methods and string kernels for authorship identification: The federalist papers case. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP-07)*, Borovets, Bulgaria, September.

Rieck, K. and P. Laskov. 2008. Linear-time computation of similarity measures for sequential data. *The Journal of Machine Learning Research*, 9:23–48.

Rieck, Konrad and Pavel Laskov. 2006. Detecting unknown network attacks using language models. In *Detection of Intrusions and Malware, and Vulnerability Assessment, Proc. of 3rd DIMVA Conference*, LNCS, pages 74–90, July.

Rieck, Konrad and Pavel Laskov. 2007. Language models for detection of unknown attacks in network traffic. *Journal in Computer Virology*, 2(4):243–256.

Sanderson, Conrad and Simon Guenter. 2006. Short text authorship attribution via sequence kernels, markov chains and author unmasking: An investigation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 482–491, Sydney, Australia, July. Association for Computational Linguistics.

Schleimer, S., D.S. Wilkerson, and A. Aiken. 2003. Winnowing: local algorithms for document fingerprinting. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 76–85. ACM New York, NY, USA.

Wang, K., J.J. Parekh, and S.J. Stolfo. 2006. Anagram: A content anomaly detector resistant to mimicry attack. pages 226–248.

Wang, K. and S.J. Stolfo. 2004. Anomalous payload-based network intrusion detection. pages 203–222.

Weber-Wulff, Debora. 2008. Softwaretest, http://plagiat.htw-berlin.de/software/2008/.

Webis at Bauhaus-Universität Weimar and NLEL at Universidad Politécnica de Valencia. 2009. PAN Plagiarism Corpus PAN-PC-09. http://www.webis.de/research/corpora. Martin Potthast, Andreas Eiselt, Benno Stein, Alberto Barrń Cedeño, and Paolo Rosso (editors).

A Appendix 1: Encoplot code

This appendix provides the listing of the implementation of the *encoplot* algorithm. At its core is a very fast implementation of the radix sort algorithm for virtually sorting the n-grams in a text without swapping any memory blocks. It is a specialization of the general radix sort algorithm. The key part is avoiding to recompute the frequencies at each step in the radix sort algorithm, and relying instead on updating those incrementally. Another key technical aspect is the use of the 128 bit unsigned integer type __uint128_t, possible with the gcc compiler on certain platforms, which allows for very good speeds up to n-grams of length 16, on 64-bit architectures, such as the common x86-64. The main code uses this virtual sorting of the n-grams sets to compute the *encoplot* data of two given files, a central part of our plagiarism detection method, as explained above.

```
//computes the encoplot data of a pair of files
#include "stdio.h"
#include "stdlib.h"
#include "string.h"
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
typedef __uint128_t tngram;
//CrG rsort
#define fr(x,y)for(int x=0;x<y;x++)

int* index_rsort_ngrams(
        unsigned char *x, int l, int DEPTH){
int NN=l-DEPTH+1;if(NN>0){
unsigned char *pin=x+NN;
unsigned char *pout=x;
int *ix=(int*)malloc(NN*sizeof(int));
int *ox=(int*)malloc(NN*sizeof(int));
const int RANGE=256;
int counters[RANGE];int startpos[RANGE];
fr(i,NN)ix[i]=i;
//radix sort, the input is x,
// the output rank is ix
fr(k,RANGE)counters[k]=0;
fr(i,NN)counters[*(x+i)]++;
fr(j,DEPTH){int ofs=j;//low endian
        int sp=0;
        fr(k,RANGE){startpos[k]=sp;
                sp+=counters[k];}
        fr(i,NN){unsigned char c=x[ofs+ix[i]];
                ox[startpos[c]++]=ix[i];}
        memcpy(ix,ox,NN*sizeof(ix[0]));
        //update counters
        if(j<DEPTH-1){
counters[*pout++]--;counters[*pin++]++;}}
free(ox);return ix;}}
#define MAXBUFSIZ 8000123
unsigned char file1[MAXBUFSIZ];
unsigned char file2[MAXBUFSIZ];
int l1,l2;

inline tngram readat(
        const unsigned char *buf,int poz){
        return *(tngram *)(buf+poz);}

int main(int argc, char ** argv){
  int depth=sizeof(tngram);
  FILE *f1=fopen(argv[1],"rb");
  l1=fread(file1,1,MAXBUFSIZ,f1);fclose(f1);
  FILE *f2=fopen(argv[2],"rb");
  l2=fread(file2,1,MAXBUFSIZ,f2);fclose(f2);
//index the ngrams
int *ix1=index_rsort_ngrams(file1,l1,depth);
int *ix2=index_rsort_ngrams(file2,l2,depth);
int i1=0;int i2=0;//merge
tngram s1=readat(file1,ix1[i1]);
tngram s2=readat(file2,ix2[i2]);
l1-=(depth-1);l2-=(depth-1);
while(i1<l1 && i2<l2){
if(s1==s2){
        printf("%d %d\n",ix1[i1],ix2[i2]);
        i1++;if(i1<l1)s1=readat(file1,ix1[i1]);
        i2++;if(i2<l2)s2=readat(file2,ix2[i2]);}
else if(s1<s2){
        i1++;if(i1<l1)s1=readat(file1,ix1[i1]);}
else if(s2<s1){
        i2++;if(i2<l2)s2=readat(file2,ix2[i2]);}}
free(ix2);free(ix1);return 0;}
```

A plagiarism detection procedure in three steps: selection, matches and "squares" *

Chiara Basile
Dip. Matematica
Univ. Bologna
P.zza di Porta S. Donato 5,
40126, Bologna, Italy
basile@dm.unibo.it

Dario Benedetto
Dip. Matematica
Sapienza, Univ. Roma
P.le Aldo Moro 2,
00185, Roma, Italy
benedetto@mat.uniroma1.it

Emanuele Caglioti
Dip. Matematica
Sapienza, Univ. Roma
P.le Aldo Moro 2,
00185, Roma, Italy
caglioti@mat.uniroma1.it

Giampaolo Cristadoro
Dip. Matematica
Univ. Bologna
P.zza di Porta S. Donato 5,
40126, Bologna, Italy
cristadoro@dm.unibo.it

Mirko Degli Esposti
Dip. Matematica
Univ. Bologna
P.zza di Porta S. Donato 5,
40126, Bologna, Italy
desposti@dm.unibo.it

Abstract: We present a detailed description of an algorithm tailored to detect external plagiarism in PAN-09 competition. The algorithm is divided into three steps: a first reduction of the size of the problem by a selection of ten suspicious plagiarists using a n-gram distance on properly recoded texts. A search for matches after T9-like recoding. A "joining algorithm" that merges selected matches and is able to detect obfuscated plagiarism. The results are briefly discussed.
Keywords: n-grams, plagiarism, coding, string matching

1 Introduction

In this short paper we aim to describe our approach to automatic plagiarism detection. In particular, we discuss how we were able to borrow and adapt some techniques and ideas recently developed by some of us for different, but related, problems such as Authorship Attribution (AA)(Benedetto, Caglioti, and Loreto, 2002; Basile et al., 2008). While some of the authors gained over the last years certain expertise in the field of AA, it is the first time we face plagiarism detection. The algorithm we are going to describe has been tailored on the *"1st International Competition on Plagiarism Detection"* (Stein et al., 2009) and does not pretend to be optimal for generic situations. Indeed we joined the competition while being completely unaware of the relevant literature, and thus the main aim of this paper is to participate to a detailed comparison of the different approaches to the contest that, we hope, will permit to enlarge the applicability of the methods and generate a profound integration and combination of different ideas.

* We thank the organizers of PAN-09 competition for the stimulating challenges and C. Cattuto and V. Loreto for bringing the contest to our attention.

2 The problem and the datasets

The contest was divided into two different challenges: external and internal plagiarism. We concentrated on the external plagiarism only. For the sake of completeness we recall the main goal (see (PAN-09-Organizers, 2009) for more details):

..given a set of suspicious documents and a set of source documents the task is to find all text passages in the suspicious documents which have been plagiarized and the corresponding text passages in the source documents.

The organizers provided a training corpus, composed of 7214 source documents and 7214 suspicious documents, each with an associated XML containing the information about plagiarized passages. A first statistical analysis shows that text lengths vary from few hundreds to 2.5 million characters while the total number of plagiarized passages is 37046. Moreover, exactly half of the suspicious texts contain no plagiarism and about 25% of the source documents are not used to plagiarize any suspicious document. The length of the plagiarized passages has a very peculiar distribution, see Figure 1: there are no passages with length in the window 6000-12000 characters, and even for long texts there is no

plagiarism longer than 30000 characters. A remarkable fact is that about 13% of the plagiarized passages consist of *translated* plagiarism.

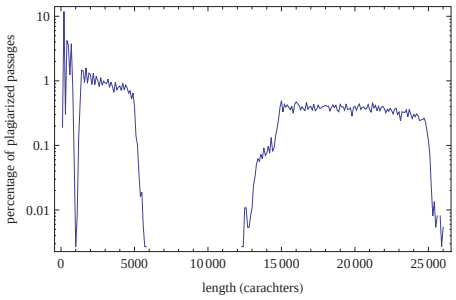

Figure 1: Distribution of plagiarized passage lengths in the training corpus.

Similarly, the competition corpus is composed of 7215 source documents and 7214 suspicious documents (obviously without any associated XML files). The length statistics are very close to those for the training corpus, see Figure 2.

The overall score is then calculated as the ratio between F-measure and granularity over the whole set of detected chars (see (Stein et al., 2009) for more details).

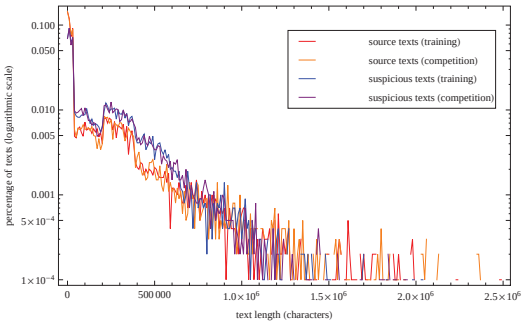

Figure 2: Text length distribution for the training corpus and for the competition corpus.

3 The method

3.1 A first selection

For each suspicious document, we first identify a small subset of source documents for a second and deeper (computationally demanding) analysis.

We start by calculating a suitable distance between each suspicious and source text. Then, for each suspicious text the first 10 source neighbors ordered according to this distance are kept for further analysis.

In order to bring computational time to a practical scale the texts were first transformed into sequences of word lengths, so that for example the sentence `To be, or not to be: that is the question` becomes simply `2223224238`. All word lengths greater than 9 were cut to 9, so that the new alphabet consists of the nine symbols $\{1,\dots,9\}$. These coded versions of the texts are on average 82.5% shorter than the original ones, and so computation times are greatly reduced.

The distance between each suspicious and source text has been computed by comparing in a suitable way the frequency vectors of the 8-grams of the coded versions. This n-gram distance used here has been proved successful in Authorship Attribution problems (Basile et al., 2008). To be more precise, after a first experiment based on bigram frequencies presented in 1976 by Bennett (Bennett, 1976), Kešelj et al. published in 2003 a paper in which n-gram frequencies were used to define a similarity distance between texts (V. Kešelj and Thomas, 2003). The distance introduced in (Basile et al., 2008) and used here should be considered as a natural development of the previous ones: once the value of n has been fixed, usually between 4 and 8, n-gram frequencies are calculated for a given text x. If we denote by ω an arbitrary n-gram, by $f_x(\omega)$ the relative frequency with which ω appears in the text x and by $D_n(x)$ the so called *n-gram dictionary* of x, that is, the set of all n-grams which have nonzero frequency in x, for any pair of texts x and y, we can define:

$$d_n(x,y) := \frac{1}{|D_n(x)|+|D_n(y)|} \sum_{\omega\in D_n(x)\cup D_n(y)} \left(\frac{f_y(\omega)-f_x(\omega)}{f_y(\omega)+f_x(\omega)}\right)$$

This is exactly the distance that has been used in (Basile et al., 2008), together with a suitable voting procedure, for approaching a two-class classification problem.

The parameter $n = 8$ for the $n-$gram distance was chosen here as a compromise between a good recall (the fraction of plagiarized characters coming from the first 10 nearest neighbors of each suspicious text is 81%) and acceptable computation times (about 2.3 days for the whole corpus). Since it was impossible to do repeated computations on the whole corpus, the optimization was done using a small subset of 160 suspicious texts and 300 source texts, suitably selected by imposing total and plagiarism length distributions comparable to those of the whole corpus.

Note that a recall of 81% is a very good result for the 8-gram distance, since the method just described has basically no hope

to recognize the 13% of translated plagiarism. Therefore, at least on the small subset of the training corpus, only about 6% of the (non translated) plagiarized passages are lost in this phase.

3.2 T9 and matches

After identifying the 10 "most probable plagiarist sources" we now perform a more detailed analysis to detect the plagiarized passages. The idea is to look for common subsequences (*matches*) longer than a fixed threshold. To this goal we need to recover some of the information lost on the first passage by first rewriting the text in the original alphabet and then using a different (and less "lossy") coding. We perform a T9-like coding: this system is typically used for assisted writing on most mobile phones. The idea is to translate three or four different letters into the same character, for example {`a,b,c`} $\mapsto 2$, {`d,e,f`} $\mapsto 3$ and so on. The symbol 0 is used for newline and blank space, 1 for all symbols other than these two and the letters of the alphabet. The new alphabet for the coded texts is therefore made up of 10 symbols: $\{0, 1, 2, \ldots, 9\}$. Note that the use of T9 "compression", which could seem strange at a first sight, can be justified by observing that a "long" T9 sequence (10-15 characters) has in most cases an "almost unique" translation in a sentence which makes sense in the original language.

The "true" matches between a suspicious and a source document are now found: from any possible starting position in the suspicious document the longest match in the source document is calculated (possibly more than one with the same length). If the length is larger than a fixed threshold and the match is not a submatch of a previously detected one, it is stored in a list.

Here, we take advantage of the choice of the T9 encoding, which uses ten symbols: for any starting position in the source document, the algorithm stores the last previous position of the same string of length 7, and for any possible string of length 7 it is memorized, in a vector of size 10^7, the last occurrence in the source file. With respect to other methods (suffix trees or sorting, for istance), in this way we can search the maximum match in the source document avoiding to compare the smaller ones.

Running this part of the algorithm on a common PC for the whole corpus of 7214 texts took about 40 hours. The threshold for the match length was fixed arbitrarily to 15 for texts shorter than 500000 characters, to 25 for longer texts.

3.3 Looking for "squares"

The previous phase gives a long list of matches for each suspicious-source pair of documents. Since the plagiarized passages had undergone various levels of obfuscation, typically the matches are "close" to each other in the suspicious texts but taken from not necessarily subsequent places in the source texts. By representing the pair of texts in a bidimensional plot, with the suspicious text on the x axis and the source text on the y axis, each match of length l, starting at x in the suspicious document and at y in the source document, draws a line from (x, y) to $(x + l, y + l)$. The result is often something similar to Figure 3: there are some random (short) matches all around the plane but there are places where matches accumulate, forming lines or something similar to a square. Non-obfuscated plagiarism corresponds to lines, i.e. a single long match or many short close matches which are in succession both in the suspicious and in the source texts, whereas intuitively obfuscated plagiarism corresponds to "squares": here the matching sequences are in a different order in the source and suspicious documents.

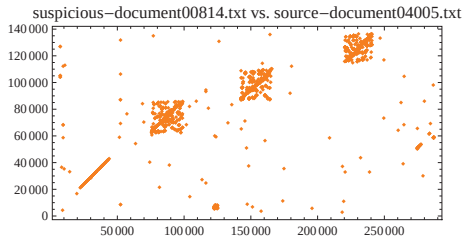

Figure 3: Orange points correspond to the position of matching chars (measured in number of chars from the beginning) between a suspicious and a source document (see top of the plot) of the training corpus. A "square" of matches corresponds to an obfuscated plagiarism.

Figure 4 is an example of what can happen when there is no plagiarism: matches are uniformly spread around the plane and do not accumulate anywhere. Obviously these are just two of the many possible settings: longer texts or the presence of "noise" (e.g. long sequences of blanks, tables of numbers...) can give rise to a much higher density of

matches, substantially increasing the difficulties in identifying the correct plagiarized passages.

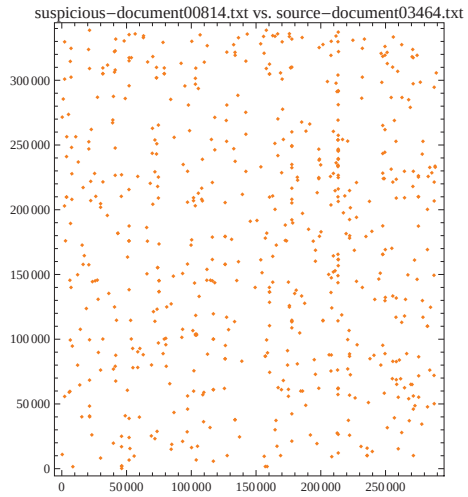

Figure 4: Orange points correspond to the position of matching chars (measured in number of chars from the beginning) between a suspicious and a source document (see top of the plot) of the training corpus. No plagiarism is present in this case.

In order to provide a single quadruple (x, y, l_x, l_y) of starting points and lengths for each detected plagiarized passage we need to implement an algorithm that joins the "cloud" of matches of each "square".

Note that the algorithm that performs this task needs to be scalable with plagiarism lengths, which can vary from few hundreds, up to tens of thousands characters.

The algorithm used here *joins two matches* if the following conditions hold simultaneously:

1. matches are subsequent in the x coordinate;
2. the distance between the projections of the matches on the x axis is greater than or equal to zero (no superimposed plagiarism) but shorter than or equal to the l_x of the longest of the two sequences, scaled by a certain δ_x;
3. the distance between the projection of the matches on the y axis is greater than or equal to zero but shorter than or equal to the l_y of the longer of the two sequences, scaled by a certain δ_y.

Merging now repeatedly the segments which are superimposed either in x or in y, we obtain some quadruples which correspond roughly to the "diagonals" of the "squares" in Figure 3. We finally run the algorithm once again using smaller parameters δ'_x and δ'_y in order to reduce the granularity from 2 to approximately the optimal value of 1. Figure 5 shows the result of the algorithm for the couple of texts of Figure 3 (blue), and Figure 6 shows the very good superimposition with the actual plagiarized passages (black), as derived from the XML file.

The procedure just described has been developed and tuned for the competition in a restricted time schedule, but it would be quite interesting to compare the efficiency of this procedure to the ones that can be achieved by using standard clustering algorithms. We plan to do this in the future.

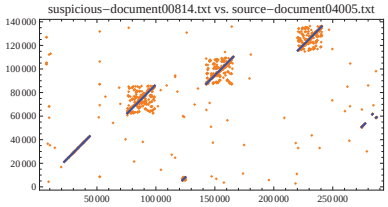

Figure 5: Detected plagiarism for the pair of texts of the training corpus indicated at the top of the plot. Single matches in orange, joined matches in blue.

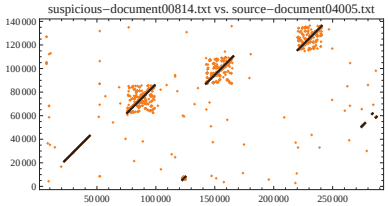

Figure 6: Plagiarized passages for the pair of texts of the training corpus indicated at the top of the plot. Single matches in orange, actual plagiarism in black. Note the perfect superimposition between the blue lines in figure 5 and the black lines here.

3.4 Tuning the parameters

The "joining algorithm" described above depends on 4 parameters: δ_x and δ_y for the first joining phase, and the rescaled δ'_x and δ'_y for the second joining phase. Our choice of the actual value in use has been dictated essentially by the lack of time and no rigorous and efficient optimization has been performed. Driven by very few trials and with some heuristic, we decided to use the following values: $\delta_x = \delta_y = 3$ and $\delta'_x = \delta'_y = 0.5$.

It is important to remark that different choices of the δ values yield to different detection results. For example, increasing their values typically results in a larger recall and in a better granularity, but also in a smaller precision. A further analysis of these dependencies could provide (in future developments) a controllable way of modifying the precision, the recall and the granularity, depending on the plagiarism detection task into consideration. A promising strategy that we plan to explore in the future consists in a dynamical tuning of these parameters according, for example, to the density of matches or to the lengths of the two texts into consideration.

The match-joining algorithm was developed using *Mathematica*© 7, and it ran on a common PC for about 20 hours on the whole data set of the competition.

4 Results and comments

The algorithm described gave the following results on the competition corpus (Stein et al., 2009):

- Precision: 0.6727
- Recall: 0.6272
- F-measure: 0.6491
- Granularity: 1.0745
- Overall score: 0.6041

The overall score is the third best result after 0.6093 and 0.6957 of the first two. We stress that the overall score drops considerably starting from the fourth position (0.3045), the fifth (0.1885), and so on. Moreover, while the winner has better results in all precision, recall and granularity, our precision and recall are better than the second, while granularity is worse.

The competition had a very tight schedule, therefore many improvements are possible. In particular, it may be that the match-joining problem can be advantageously formulated in the framework of Hidden Markov Models. Also, it would be worth to see how standard clustering algorithms perform in this case.

We are eager to compare techniques and ideas with the other participants of the contest.

References

Basile, C., D. Benedetto, E. Caglioti, and M. Degli Esposti. 2008. An example of mathematical authorship attribution. *Journal of Mathematical Physics*, 49:125211–125230.

Benedetto, D., E. Caglioti, and V. Loreto. 2002. Language Trees and Zipping. *Physical Review Letters*, 88(4):048702–1, 048702–4.

Bennett, W.R. 1976. Prentice-Hall, Englewood Cliffs, NJ.

PAN-09-Organizers. 2009. Proceedings of PAN-09.

Stein, B., M. Potthast, A. Eiselt, P. Rosso, and A. Barrón-Cedeño. 2009. http://www.webis.de/pan-09/competition.php.

V. Kešelj, F. Peng, N. Cercone and C. Thomas. 2003. n-gram-based author profiles for authorship attribution.

Finding Plagiarism by Evaluating Document Similarities

Jan Kasprzak, Michal Brandejs, and Miroslav Křipač
Faculty of Informatics, Masaryk University
Botanická 68a, 602 00 Brno, Czech Republic
kas@fi.muni.cz, brandejs@fi.muni.cz, kripac@fi.muni.cz

Abstract: In this paper we discuss the approach we have used for finding plagiarized passages of text during the PAN'09 plagiarism detection competition. We describe the existing anti-plagiarism system we use in the Czech National Archive of Graduate Theses. We then discuss the modifications to this system which have been necessary in order to fit the results to the competition rules. We also present a performance data of the described system, and the possible improvement for our production systems, which result from the code written for the PAN'09 competition.
Keywords: Plagiarism, Similar Documents, Document Overlap, Distributed Computing, Parallelism

1 Background

At Masaryk University, the study administration is being supported by the Masaryk University Information System (IS MU, 1999–2009). The integral part of this system is the distributed document storage, used by various subsystems, including e-learning agendas, archive of graduate theses, etc. The document storage is a feature-rich subsystem with some properties similar to the common file systems (hierarchical directory-based storage, object as a stream of bytes, etc.).

Some features are quite unique to this document storage: the storage supports multiple versions of the same document (e. g. DOC, PDF, and plain text) as a single entity, automatic conversion between the file formats (including OCR of the scanned PDF files, generating thumbnail images for the picture files such as JPEG, etc.), distributed replication with strong checksums, wide set of access rights, etc.

Since August 2006, the storage subsystem supports also finding similarities between documents, in order to assist with discovering plagiarism. The first version of the system has been a prototype coded in Perl with the DBI interface, using Oracle database as a metadata back-end.

In early 2008, the system has been replaced by the custom distributed solution, outlined in our earlier work (Kasprzak et al., 2008), distributed aspects of which we further discuss in (Kasprzak, Brandejs, and Brandejsová, 2009). The same underlying system is currently in use also in the Czech National Archive of Graduate Theses (Theses.CZ, 2008–2009). This distributed system has been used as a basis for solving the external plagiarism task in the PAN'09 competition (PAN'09, 2009).

2 General Approach

There are several possible approaches for finding similarities in a given base of documents, some of them are discussed and compared in (Monostori et al., 2002). The IS MU and Theses.CZ anti-plagiarism system currently uses the approach similar to the one described by (Finkel et al., 2002).

2.1 Tokenization

We use words as the base units which we handle. In order to overtake the need of stemming (which would bring the dependence on the particular language to the system) and also to handle some means of obfuscation of the Czech language we translate the words to the US-ASCII by stripping off diacritics[1], and ignoring short words, which in the Czech language are mostly prepositions. We do not use stop words of any kind.

[1] For example: `tučňák` → `tucnak`

2.2 Creating the Index

The tokens (words) are then joined into *chunks*. We use overlapping sequences of several words, and we have generally had better results with shorter chunks of about 4 to 6 words than the larger chunks as used by (Broder, 1997) or (Finkel et al., 2002). The chunks are then hashed by a hash function (for our purposes, the value range of 28 to 32 bits seems to be sufficient, even though the probability of hash function collisions is higher with smaller number of bits). There are no special requirements to the hash function itself—we have used the hinghest n bits of the MD5 hash (Rivest, R., 1992).

The mapping of document ID to the sequence (or a set) of hash values is then constructed, and an *inverted index* mapping the hash value to the sequence of document IDs is computed from it.

2.3 Computing the Similarities

This inverted index is then used for finding similarities in a given document base. For now, the system computes only a single numeric value for each pair of documents in a given set. This value represents the number of chunks which these two documents have in common (not taking the possible hash function collisions into the account).

The most common use case is to discover which documents are similar to the given document (e. g. a newly imported thesis). We postpone the computation of the actual similar passages of the text to the time when the user wants to see them.

2.4 Hardware Configuration

The IS MU and Theses.CZ systems use a common document base of about 1,300,000 documents. The anti-plagiarism software runs on a cluster of 45 non-dedicated PCs with various dual-core CPUs (AMD and Intel), and a dedicated server with Oracle database for storing the computed results. Recomputing the similarities across the whole document base takes about three hours, most of which is spent by importing the results to the database. The incremental run (after adding some more documents to the system) then takes 12 to 25 minutes depending on the overall system load (the servers in the cluster have other tasks to do besides computing the similarities).

3 The PAN'09 Competition

The requirements for the PAN'09 competition were quite different to what we currently do in IS MU and Theses.CZ systems. The main difference was that we should not only find the similarity between the documents, but also to show exactly where the similarities are. Another important rule to consider was the granularity measure, which had been very strong, especially in the earlier version of the rules.

The first approach we wanted to try was simply to import both the suspicious documents and the source documents to the IS MU system, and let it find the similarities. This would have been a very straightforward solution, requiring no programming except post-processing the results on our side. Further examination of the data led us to the opinion that we can do better by modifying our software to match the requirements of the competition.

We have taken the core modules of our system and modified them to run on a single multi-core computer. Both the source corpus and the set of the suspicious documents were quite small (relative to what we have to handle in our production systems), so the relevant data could fit to the RAM of a single mid-range server.

3.1 Rich Tokenization

We have modified the tokenization process to get not only a list of words of a given document but also for each word to include the position of the word in the document expressed as two distinct numbers: firstly as the offset of the first character in the word from the beginning of the document, and secondly the count of the (non-ignored) words discovered so far in this document.

Using this additional data, we can construct the document chunks with the extra attributes: the offset of the first and last character of the chunk[2], and the sequence number of that chunk[3].

The tokenization process has been further modified to include all the Unicode alphanumeric characters as word characters. In our production systems, we do not count digits

[2] The offset of the last character in a chunk is computed from the offset of the last word of that chunk and the length of that word.

[3] The sequence number of that chunk is equal to the sequence number of the first word of that chunk.

as word characters, because a common case of plagiarism of student seminar works e. g. in biology is to take the work of the other student, and simply change the numbers in measurements.

3.2 Computing the Inverted Index

The computation of the inverted index (i. e. mapping the chunk hash value to the list of document IDs) has been modified to contain the additional data (the chunk sequence number, and range of characters which the chunk covers). In order to allow the case when a single plagiarized passage has more than one possible source passage in the same source document, we have also allowed the repeated occurences of the same chunk hash value within the same source document.

3.3 Finding the Similar Document Pairs

Using this enhanced inverted index, we can now tokenize and evaluate the suspicious documents in order to find similarities. For each suspicious document, we split it to the chunks, and look up their hash values in the index. This gives us the list of the documents, and positions of the chunks in them[4]. We can now use a cut-off value, and further handle the document pairs with at least this value of common chunks.

For the competition itself, we have used the cut-off value of 20 chunks, which together with 5-word chunks gives us a minimum of 24 common words that we consider a similarity between documents.

4 Discovering Similar Passages

In order to fulfill the requirements of the competition, we have not only to compute the similarity of the document pairs, but also to show exactly which passages are similar. The computation against an inverted index gives for each suspicious document the list of possible source documents, and for each source document a list of common chunks. With each common chunk we have its sequence number and characters range from the suspicious document, and (possibly more than one) sequence number and character range from the source document.

[4]Remember that a chunk from the suspicious document can be identical to several chunks in one source document.

Using this data, we can further narrow the scope, and keep only the larger similar passages. How to compute the similar continuous passages from the data is not clear. Our approach is to consider only "dense enough" intervals of the suspicious document, which also map to the "dense enough" intervals of the source document.

Since the computation has to be done both from the point of view of the suspicious document and the source document, for the algorithm it does not matter which document we are currently looking at. We will further use the terms D_1 and D_2.

We have considered only intervals of chunks of D_1 matching the following criteria:

1. The first and the last chunk of this interval are present in the D_2.
2. The interval should have at least 20 (possibly overlapping) chunks, which are also present in D_2.
3. Between each two adjacent chunks from the interval which are also present in D_2, there should be at most 49 chunks which have no matching chunk in D_2.

This interval we will hereby call a *valid interval*.

For example, when the suspicious document we have chunks numbered

$$50, 100, 150, ..., 950, 1000$$

which are all present in a particular source document, we consider the interval 50–1000 to be a valid interval.

We further process only those valid intervals, which also map to the valid interval in some source document. We consider part of the suspicious document covered by the chunks from this valid interval to be a plagiarized passage. The plagiarized passage can be computed using the algorithm, described in the in the next subsection:

4.1 Algorithm: Valid Intervals

The input of the algorithm is a list of pairs (chunks ID in D_1, matching chunk ID in D_2). If one chunk in D_1 maps to more than one chunk in D_2, the list has more than one entry for this chunk.

1. Set the local variable `depth` to 0.

2. Sort the list of pairs by the chunk ID in D_1.
3. Split the list to the largest possible valid intervals in D_1, ignore the chunk ID pairs which are not present in any valid interval.
4. If there is only one valid interval covering the whole input list, increase the `depth` variable by 1.
5. If the `depth` variable is equal to 2, return the whole range of chunk IDs as the resulting plagiarized passage.
6. For each valid interval, do the following:
   (a) Create a new list of chunk ID pairs as (chunk ID in D_2, chunk ID in D_1), where the chunk ID in D_1 is from the current valid interval.
   (b) Set the variable `depth` to 1.
   (c) Rerun recursively the algorithm, starting from the step 2.

4.2 Postprocessing

During the postprocessing phase, we remove possible overlapping passages for each suspicious document, keeping only the largest passage from the set of overlapping ones. This is to meet the nature of the competition data. In the real system, we would like to keep all the possible similarities, even the overlapping ones.

5 Practical Results

5.1 Implementation

The existing implementation of IS MU and Theses.CZ anti-plagiarism system uses a mixture of C and Perl code (C for performance-critical code like generating an inverted index of chunks or searching this index, Perl for feature-rich parts including the communication with the SQL database and generating chunks from the text files). It is portable to any 64-bit POSIX system. For the PAN'09 competition, it had to be only slightly modified, and a new valid interval evaluation and postprocessing system has been written. We have worked on the PAN'09 competition for four days, and additional half a day after the deadline has been extended. We did not do any fine-tuning against the evaluation formula, based on the development corpus.

5.2 Performance

The computation for the PAN'09 competition has been run on a single mid-range server: dual Xeon E5472 (3.0 GHz, total of 8 cores), 64 GB RAM, and a RAID-10 array of eight 15k RPM disks. The system runs Fedora Linux with the Ext4 filesystem. Most parts of the computation have been parallelized on a document-by-document or a document pair-by-pair basis.

Generating the inverted index from the corpus of 7124 source documents took 34 minutes. Finding the matching chunk pairs against the 7124 suspicious documents took about 38 minutes. Computing the maximum valid intervals from this data, postprocessing and generating the XML output files took 2 minutes. Because of the relatively big RAM available in the server, the computation was mostly CPU-bound.

5.3 The Benefit of Valid Intervals

Computing the valid intervals instead of just the similarity between the documents can surprisingly enough be a win not only from the viewpoint of getting a more meaningful data, but also from the performance standpoint: In our existing systems, most of the time during the computation is spent by inserting the similarity data to the database. Using the valid intervals can greatly reduce false-positives, and thus the number of rows (document pairs) which we need to insert to the database:

In the development corpus we had about 576,000 document pairs, which had at least 20 chunks in common. Actually looking at those chunks and keeping only those which form a valid interval both in the suspicious document and in the source document reduced this number to about 18,000 document pairs with 47,000 similar passages.

As it can be seen from the previous subsection, the cost of this step is relatively insignificant when compared to the other steps, even though we have implemented this step purely in Perl.

6 Conclusion

We have implemented the system which can find similar documents relatively fast, given a multi-core machine. The same approach can even be used on a cluster of computers, which can provide a significant benefit of a distributed memory for large document sets.

This system can detect even moderately obfuscated similar passages in a given document base. In the development corpus, we have found a big number of similar passages, which have not been annotated in the development data as plagiarized[5].

In the competition itself, we had the highest recall ratio[6] amongst all the teams (69.67 %, the second highest was 65.85 %). Given our relatively poor precision ratio (we were 7th in this parameter with the precision of 55.73 %, the highest precision reached was 74.73 %), we may even have found the biggest number of *all* the similar passages, including those not generated by the machine plagiarist.

Our system currently cannot handle translations (although the work is in progress to handle a plagiarism between Czech and Slovak languages, which are quite similar in structure and vocabulary). We also cannot handle highly obfuscated text, which to us non-native English speakers does not even look similar to the source text[7].

We have seen that evaluating the particular similar passages instead of just the overall document similarity can be a significant improvement, so this result from the competition is an enhancement to incorporate back to our production systems.

Acknowledgements

We would like to thank the organizers of the PAN'09 competition. Taking part in the competition has been a very enlightening experience for us.

References

Broder, A.Z. 1997. On the resemblance and containment of documents. In *Compression and Complexity of Sequences 1997. Proceedings*, pages 21–29, Jun.

Finkel, Raphael A., Arkady Zaslavsky, Krisztián Monostori, and Heinz Schmidt. 2002. Signature extraction for overlap detection in documents. In *ACSC '02: Proceedings of the twenty-fifth Australasian conference on Computer science*, pages 59–64, Darlinghurst, Australia. Australian Computer Society, Inc.

IS MU. 1999–2009. Masaryk University Information System. `http://is.muni.cz/`.

Kasprzak, J., M. Brandejs, and J. Brandejsová. 2009. Distributed aspects of the system for discovering similar documents. In *ITA 09: Proceedings of the Third International Conference on Internet Technology and Applications*.

Kasprzak, J., M. Brandejs, M. Křipač, and P. Šmerk. 2008. Distributed system for discovering similar documents. In *ICEIS 2008: Proceedings of the Tenth International Conference on Enterprise Information Systems, Vol. DISI – Databases and Informations Systems Integration*, pages 437–440. INSTICC (Institute for Systems and Technologies of Information, Control and Communication), Setúbal, Portugal.

Monostori, Krisztián, Raphael A. Finkel, Arkady B. Zaslavsky, Gábor Hodász, and Máté Pataki. 2002. Comparison of overlap detection techniques. In *ICCS '02: Proceedings of the International Conference on Computational Science-Part I*, pages 51–60, London, UK. Springer-Verlag.

PAN'09. 2009. 1st International Competition on Plagiarism Detection. `http://www.webis.de/pan-09`.

Rivest, R. 1992. RFC1321: The MD5 Message-Digest Algorithm. `http://www.rfc-editor.org/rfc/rfc1321.txt`.

Theses.CZ. 2008–2009. Czech national archive of graduate theses. `http://theses.cz/`.

[5] As an example, the suspicious document 02010, characters 198,025 to 200,472 correspond to the source document 06059, characters 429,098 to 431,429 almost exactly. The development corpus can be downloaded from the competition web site.

[6] Refer to the competition web site for the exact description of the evaluation criteria, including the terms *recall* and *precision*.

[7] An example from the development corpus is the suspicious document 00002, characters 618,098 to 618,798, which are annotated to be similar to the source document 02400, characters 33,963 to 34,664.

Tackling the PAN'09 External Plagiarism Detection Corpus with a Desktop Plagiarism Detector*

James A Malcolm
University of Hertfordshire
College Lane, Hatfield, Herts
j.a.malcolm@herts.ac.uk

Peter C R Lane
University of Hertfordshire
College Lane, Hatfield, Herts
p.c.lane@herts.ac.uk

Abstract: Ferret is a fast and effective tool for detecting similarities in a group of files. Applying it to the PAN'09 corpus required modifications to meet the requirements of the competition, mainly to deal with the very large number of files, the large size of some of them, and to automate some of the decisions that would normally be made by a human operator. Ferret was able to detect numerous files in the development corpus that contain substantial similarities not marked as plagiarism, but it also identified quite a lot of pairs where random similarities masked actual plagiarism. An improved metric is therefore indicated if the "plagiarised" or "not plagiarised" decision is to be automated.
Keywords: Ferret, n-grams, plagiarism detection, similarity

1 Introduction

In this paper we describe how we approached the challenge of the PAN'09 Plagiarism Detection Competition using the Ferret plagiarism detection software. We outline Ferret's strengths in normal use, highlight the difficulties we had in using Ferret for the competition task, and describe the results of the improvements that we made as a result of entering the competition.

The "external plagiarism analysis" task of the PAN'09 Plagiarism Detection Competition (International Competition on Plagiarism Detection, 2009) is an example of category 1 plagiarism (Lyon and Malcolm, 2002), as we have the source(s) in our hand. This suggests that Ferret is the tool for the job.

Ferret (Lyon, Barrett, and Malcolm, 2004; Lyon, Malcolm, and Barrett, 2005) is a tool that (when used on student work) is primarily good for detecting collusion rather than plagiarism (though it has been extended to generate search terms to drive an Internet search (Malcolm and Lane, 2008b)). It is a desktop plagiarism detector, which means that it is fast and interactive. It has to be fast, because a human is waiting for the results, and because it is interactive, human input is available and appropriate: after all, plagiarism is an academic judgement not something that can be measured by a machine (Flint, Clegg, and Macdonald, 2006; Lyon, Barrett, and Malcolm, 2004). Before tackling the competition we had done little to automate the decision making process: "is this plagiarism or not"; Ferret tells its user *which* pairs to look at (and helps in reviewing those pairs) but leaves the actual decision has to him or her. There is therefore no need for the software to draw a dividing line – a ranked list is sufficient.

The competition did highlight what we knew to be Ferret's strengths and weaknesses: we came second on recall, but precision was poor as Ferret is *too* fine grained in its identification of similarities.

2 Ferret's Strengths

Assuming, as in the competition, that we already have the sources of all the copying, then there are two tasks in identifying plagiarism in a collection of documents: finding which pairs of documents to look at and (once a pair has been selected for further examination) finding the blocks of matching text.

In the case of the competition there *appear* to be $(7214)^2$ comparisons to be made. In the more general case usually considered by Ferret, every file needs to be compared with every other, giving $\frac{n \cdot n-1}{2}$ comparisons (which for source and suspicious files together

* We thank Bob Dickerson who developed the core from which the original Ferret code was developed.

is 104,076,378 pairs). But it is important to note that the way Ferret works these comparisons are not made explicitly; as the documents are read by Ferret it creates a data structure which enables the most similar pair of documents to be selected *without* making a direct comparison of those two documents. To be more specific, it remembers every three word sequence (triple) that it has seen, and which input files that triple appears in. Similar files are those with the largest number of common triples. This simple approach is what makes Ferret fast.

The Ferret user interface then allows the operator to display the similar documents side by side with the similar text highlighted. For large documents, it can take as long to display one pair as it does to find all the similarities in the set (we mention this to highlight the speed of the first phase, rather than as a deficiency of the user interface).

There is no specific support in Ferret for finding the sections of a source that has been copied. This is done by the operator (although he or she can click on any one of the matching triples to find where in the two compared documents it appears).

We expect to look at the most similar pairs (in descending order of similarity) and stop when we judge that plagiarism is no longer occurring. In effect this is a corpus specific threshold (partly mitigating the problem mentioned in section 3.2).

3 Adapting for the Competition

The problems that have to be solved in order to use Ferret for the competition relate firstly to the large volume of data to be examined and secondly to the difference between how we would normally use Ferret and how the competition is run.

3.1 Scale

To deal with the large volume of data in the competition, we had to divide the input into batches that were processed in turn as we did not have a machine with sufficient RAM to deal with all the data in one go. We estimate that 32GB would be enough; our machine was 9GB (for normal use, Ferret runs in 512KB or less). If batching is necessary, it is most efficient if half the available memory is used for source documents, and half for suspicious documents.

If M is the available memory, the number of batches of source documents, N_O, will be $\frac{2|O|}{M}$ (where $|O|$ is the total size of all the source documents in the corpus). The number of batches of suspicious documents, N_U, should be $\frac{2|U|}{M}$. The number of runs, $N_O \cdot N_U$, will thus be $\frac{4|O||U|}{M^2}$ which is quadratic in corpus size; doubling the memory available will make the system four times faster.

3.2 Automation

We needed to automate the decision between "plagiarised" and "innocent similarity". At present Ferret supports two possible metrics: a count of the number of common triples, and the Jacquard coefficient, 'R'.

Some of the files are very big, but these are mixed with quite small files so our current metric does not work very well. The copied chunks vary from a couple of sentences up, but it is the huge size of some of the files that causes the problem, because the number of randomly matching triples in a huge file is bigger than the size of one of the smaller copied chunks.

Examining results for the first 100 suspicious files in the Development corpus we found that we should take the 50 most similar pairs to catch all the suspicious files where artificial plagiarism had been introduced.

Ferret picks out many very small similarities. Eliminating these "accidental" similarities (common triples, typically isolated) was a problem that we had to address with code if we were to have success in the competition. It was not a problem we had seen before because of the way we use Ferret.

3.3 Improvements

Our submission was the *first* run of the complete system.

We later revised it to take some account of the order of triples in the source document when deciding if matching triples in the suspicious document are part of a matching block or just a random match. This code is quite slow, and there are now a lot of parameters that can be fiddled to change how well Ferret would perform in the competition:

- How many triples matching is considered too small to be worth considering as input to the second phase: currently less than 50 (or $R < 0.007$).

- How many documents to keep in the "most similar pairs" list: 5 on the first (submitted) run, but considering 50.
- The unmatched gap between matching sections that can be merged: 1 in the first run; considering up to 4.
- How many sections before or after the current section we can jump when merging: no restriction in first run; considering a range from 5 before to 10 after.

4 How the system operates

4.1 Identifying Similar Pairs

First we run Ferret on the complete corpus. A bash script `make-input-document-list` that creates an input file for the `ferret -f` (definition file) option. Several copies of the `make-input-document-list` script are combined in a script that does multiple runs of Ferret to do (small) groups of suspicious files against (largish) groups of source files to produce a set of output files.

4.2 Identifying Sources

We read the output files generated by step 1 to select the likely sources for a given suspicious document. This uses a ruby script `process-output` that runs `ferret -x` to produce an XML file highlighting the similarities in a particular suspicious-source pair where the number of matches and/or resemblance metric meets hard coded (but easy to change) constraints. This produces a set of XML files (one for each `ferret -x` run); these are scanned by another ruby script: `read-ferret-xml-files` to produce output in the required XML format. Formatting the results to meet the competition requirements raised a minor difficulty that the source offset required was not available in our system; fortunately it did not appear to be part of the evaluation metric.

5 Resource Analysis

Tackling more than about 500 files at a time on a 1GB laptop led to it thrashing. On a 9GB server, about 5000 files at a time could be dealt with comfortably.

For the development corpus, the output from Ferret was divided into 146 files: each file has results for about half of the source files (numbers 1-3999 or 4000-7214) and about 100 suspicious files. As explained above, this is not the best way of organising the data, but it was initially easier to test a few suspicious files at a time.

Step 1 produces a *lot* of output. As an indication of the scale of the problem, running the Unix wc (word count) utility on the complete set of output files took about 37 minutes, involving 7 minutes CPU time. The average size of the files is around 45MB, and the total size about 6.5GB.

This emphasises the quadratic nature of plagiarism detection: every suspicious file has to be checked against every possible source file. In the case of the competition this is 52,041,796 comparisons. As mentioned earlier, Ferret usually compares every file with every other, but fortunately we had already implemented a grouping facility (it has several other applications) whereby files in a group are *not* compared with each other, but only with files in other groups. We should have filtered out the least similar pairs *before* generating the output from phase 1, as the set of phase 1 output files is considerably larger than the set of suspicious documents. In normal use, Ferret displays a list of the most similar pairs; the user only looks at the most similar and because the rest of the list is in memory there is little extra cost involved.

6 Results

Here now are our observations on running the system we built around Ferret on the development corpus (we do not yet know the "correct answers" for the competition corpus).

We group the pairs into 5 types depending on the kind of artificial plagiarism:

- raw plagiarism (without obfuscation)
- low obfuscation
- high obfuscation
- plagiarism by translation
- no artificial plagiarism

We plotted the value of R for each of these types: figure 1 shows (for first 500 suspicious documents of each type in the development corpus) how documents which were identified as plagiarised (with various degrees of obfuscation) differed from those where no similarity was intended.

We see immediately that Ferret is (as expected) ineffective at detecting plagiarism by translation (the line *on top* of the x-axis), so this is left out of the later graphs.

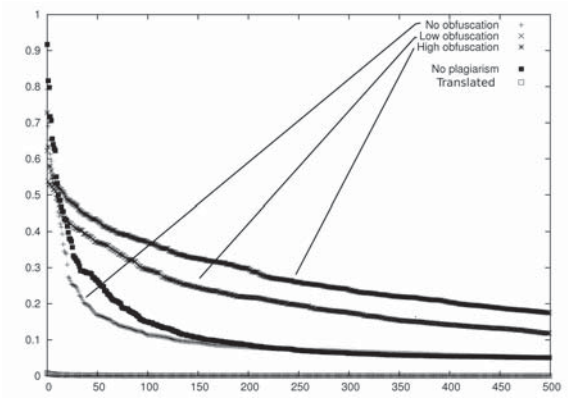

Figure 1: R for the first 500 pairs of each type

Suspicious	Source	R	Cause
00569	04692	.533	editions
01302	05069	.917	read me
01656	04163	.501	editions
01756	03972	.662	editions
01862	03766	.640	editions
02483	05054	.550	fragments
02730	01431	.629	fragments
04740	04973	.717	editions
05096	00620	.622	fragments
05959	06555	.706	editions
05964	06148	.816	editions
06188	05357	.798	plagiarism?

Table 1: Most similar non-plagiarised pairs

We also note some very high values for R, and not just where there is introduced plagiarism. Some of the pairs where there was no artificial plagiarism showed very high similarity using Ferret; there are some very high values of R before the graph flattens off. In total 49 pairs (in the development corpus) have $R > 0.5$. Twelve of these are pairs where no plagiarism is alleged. We looked at each of these pairs in detail, and present the results in Table 1.

The worst case was suspicious document 1302 which had $R = 0.91$ when compared to source document 5069. It turned out that these were both Project Gutenberg (1971-2009) "READ ME" documents with no other text. I guess this could be viewed as an accident on the part of the compilers of the corpus (Potthast, Martin *et al.* (editors), 2009). Some of the other similarities are more interesting, such as $R = 0.80$ between "The Impossibles" and "Out Like a Light", both by the same author. Despite the considerable number of small changes between the two documents, Wikipedia Authors (2009) suggest this is a re-publication of the *same* work under a new title.

Most of the high similarities in documents not alleged to be plagiarised were different editions of the same work, such as a volume which is repeated (with numerous spelling corrections) in the collected works or a "Second Edition Revised and Enlarged" with obviously a considerable overlap with the first edition.

In most of these 12 cases, we have different editions of the same work, or different collections containing most of the same material. A few are incomplete fragments, possibly arising from the way in which Project Gutenberg used to be distributed, so it is hard to tell how the similarity to other fragments came about.

We would presume that R values below 0.5 also indicate similar situations, as previous work has shown that $R > 0.04$ is the limit for "accidental similarity" but maybe a larger value is appropriate for this corpus.

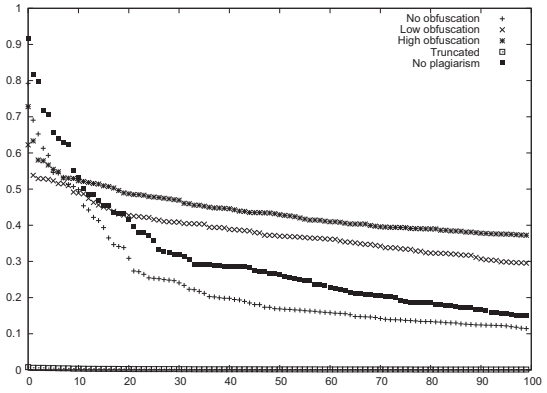

Figure 2: R for the first 100 pairs of each type

In figure 2 (where R is plotted for only the first 100 pairs of each type), it can be seen more clearly that the graphs for obfuscated plagiarism are higher than for raw, maybe because the obfuscated cases are longer: the average amount of source material in suspicious documents containing raw (un-obfuscated) plagiarism was 20,628 whereas for low and high obfuscation the average lengths were about 50% higher at 30,402 and 33,330 respectively.

Figure 3 compares R for the four types across the full range of pairs. Here the x-axis is a percentage of the total number of

pairs of the type and the most similar pairs of each type ($R > 0.4$) have been omitted. The big difference between plagiarised and non plagiarised is evident, but also we note that at the RHS of the diagram these lines merge. This is because the influence on the R metric for small pieces of plagiarism in large documents is rather too small.

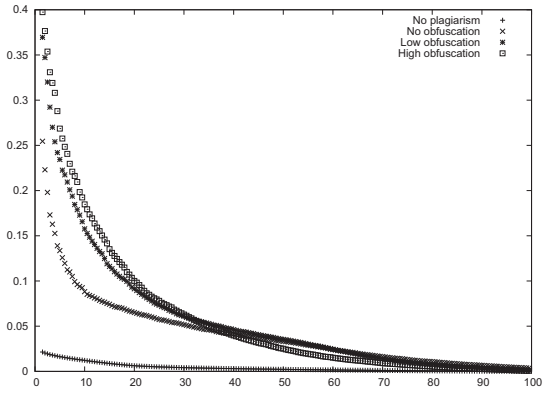

Figure 3: R for all the pairs of each type

7 Conclusions and Future Work

Producing a corpus with no similar text except that which has been added deliberately is hard: there is far too much duplicate data on the Internet to make it easy. It is for this reason that this paper concentrates on the Development corpus, as we know which parts of it are *supposed* to be plagiarised. However the competition organisers have done an excellent job in encouraging research.

The competition has clearly shown us that a better metric than R is needed. As we have suggested before (Malcolm and Lane, 2008a), a metric that takes account of the order of the similar features looks promising. Calculating the longest common subsequence of triples would probably work well, but is computationally costly; we want to take care not to slow Ferret down.

We need to develop better approaches to spanning gaps caused by obfuscation, especially in very long files as our current algorithm can still get two isolated triples that happen to be in the right order, for example a b c X k l m turning into a 7 word match, which is probably long enough to appear in the output. We should also optimise the other parameters listed in subsection 3.3.

Ferret's strength is its speed: we were able to upgrade our machine from 9 to 32GB of RAM, so can now process the entire competition corpus in a single ferret run of 1h42m. This works out at an effective rate of comparison of 50,000 pairs per second. The input is read at 450kB/s (this includes *all* ferret's processing, including writing out the very large results file).

References

Flint, Abbi, Sue Clegg, and Ranald Macdonald. 2006. Exploring staff perceptions of student plagiarism. *Journal of Further and Higher Education*, 30(2):145–156, May.

International Competition on Plagiarism Detection. 2009. `http://www.webis.de/pan-09/competition.php`.

Lyon, Caroline, Ruth Barrett, and James Malcolm. 2004. A theoretical basis to the automated detection of copying between texts, and its practical implementation in the ferret plagiarism and collusion detector. In *Plagiarism: Prevention, Practice and Policies Conference*, June.

Lyon, Caroline and James Malcolm. 2002. Experience of plagiarism detection and prevention in higher education. In *Proceedings of the World Congress, Networked Learning in a Global Environment: Challenges and Solutions for Virtual Education*. ICSC-NAISO Academic Press.

Lyon, Caroline, James Malcolm, and Ruth Barrett. 2005. The ferret copy detector: finding similar passages in large quantities of text. In *Submitted to the 43rd Annual Meeting of the Association for Computational Linguistics*.

Malcolm, J.A. and P.C.R. Lane. 2008a. An approach to detecting article spinning. *Proceedings of the Third International Conference on Plagiarism*.

Malcolm, James A. and Peter C. R. Lane. 2008b. Efficient search for plagiarism on the web. In *Proceedings of i-TCE 2008*.

Potthast, Martin *et al.* (editors). 2009. PAN Plagiarism Corpus PAN-PC-09. `http://www.webis.de/research/corpora`.

Project Gutenberg. 1971-2009. `http://www.gutenberg.org/`.

Wikipedia Authors. 2009. `http://en.wikipedia.org/wiki/Mark_Phillips_(author)`.

Putting Ourselves in SME's Shoes: Automatic Detection of Plagiarism by the WCopyFind tool

Enrique Vallés Balaguer
Private Competitor
enriquevallesbalaguer@gmail.com

Abstract: Thanks in part, to the large amount of information circulating today on the Internet, unfortunately, the plagiarism has become a very common practice, up to become one of the biggest problems of today's society. One of the most affected sectors by the plagiarism are small and medium entreprises (SME's), which are daily victims from their competitors. Finding a system able to detect plagiarism in texts, has become a major goal for the interests of SME's, which are forced to solve the problem through the tools available on the web. In this paper we analyze the results obtained in the PAN'09 competition with the WCopyFind tool.
Keywords: Plagiarism detection, WCopyFind

1 Introduction

Internet is one of the greatest advances in history in the area of communication. Thanks to (the) Internet, you can have immediate access to information, regardless of the distances. However, the easy access to information, has increased the number of plagiarism cases.

Within the business area must be emphasized the importance of the automatic plagiarism detection for SME's. For SME's is vital to know if their proposals, products, ideas, etc, have been plagiarized by competitors. To solve this problem, the companies have mainly to rely on the software available on the web. In this paper, we attempt using the software WCopyFind (Dreher, 2007) developed in the University of Virginia.

2 Plagiarism detection for SME's

SME's build web pages to enter information about themselves, advertise their products, etc..., to approach (to) the consumer. But the information on the web is also visible for the competitors. When a company launches a new tool, this is discovered by competitors within a few hours or days. But, there are companies that use this information to copy. The automatic plagiarism detection aims to try to find an automated approach that is able to locate fragments of texts suspects of plagiarism.

Currently the automatic plagiarism detection is divided into two different branches. By one side, is the *external plagiarism analysis*, which requires a set of original sources from which seeking possible plagiarized fragments in suspicious texts. Within this branch, there are methods developed with the intention to locate fragments suspected of plagiarism through search strategies.

Given the large amount of information available at present, comparing a suspected document with all the available ones is a virtually unmanageable task. Therefore, emerged the *intrinsic plagiarism analysis*, tries to rely on the suspected document. Its intention is to capture the style and the complexity of a document with the aim of finding unusual fragments that are candidates to be instances of plagiarism (Barrón-Cedeño and Rosso, 2009).

2.1 WCopyFind

WCopyFind[1] is a software developed in 2004 by Bloomfield at the University of Virginia. To detect suspicious fragments of plagiarism, WCopyfind conducts a search through the comparison of n-grams.

Since WCopyfind works with n-grams, language is not important and matches are

[1] http://plagiarism.phys.virginia.edu/

n-gram	Precision	Recall
4	2.05 %	66.34 %
5	11.34 %	59.08 %
6	17.85 %	57.06 %

Table 1: Training Phase

readily identified from the candidate documents submitted for analysis (Dreher, 2007).

3 Corpus

The PAN'09 corpus which refers to the External Plagiarism Analysis task, consists mainly of documents in English, in which you can find any type of plagiarism.

There are a total of 7,214 suspicious documents, which may contain plagiarized fragments from one or more original documents or do not contain any plagiarized fragment at all. On the other hand, the number of original documents that constitute the corpus is 7,215.

4 Results

Due to the fact that the WCopyFind tool allows the user to select the size of the n-grams, before carrying out the analysis on the competition corpus, we have made several experiments with training corpus to find the appropriate size of the n-grams. Table 1 shows the results for each one of the experiments. We can highlight several interesting points. By one side it is noteworthy that contrary to other language engineering tasks, we must stress that the obtained precision is smaller than the obtained recall.

Another interesting fact observed in Table 1 is that, how much smaller size of n-grams is, the smaller is the precision. However, it happens all the contrary to the measure of recall, that is, the smaller the n-grams, the greater is the recall. This is because, the smaller are the n-grams, the greater is the possibility of finding similar fragments in plagiarized documents. In (Barrón-Cedeño and Rosso, 2009), the authors analyzed this fact, and they showed that the probability of finding common n-grams in different documents decreases as n increases.

Finally, we have taken the decision that the best size for the n-grams was *hexagrams*, because there is no great loss with to respect of recall and it has the best result in precision.

Software	Precision	Recall
WCopyFind	1.36 %	45.86 %

Table 2: Final results obtained

Table 2 shows the results that we have obtained. From the results, we can noting that the results are not good, especially in terms of precision which is very low.

5 Conclusions and further work

Unlike most areas of the language engineering, in the automatic detection of plagiarism, the precision is lower than the recall. This is because it is very likely to find similar fragments between two documents, although these are not plagiarized fragments. For a future work, it would be interesting search for a automated approach to reduce the space of search before conducting the search based on the comparison between n-grams. In (Barrón-Cedeño, Rosso, and Benedí, 2009), the author proposed the reduction of the space of search on the basis of the Kullback-Leibler distance.

In this paper we tried to put ourselves in a SME's shoes and in its need of detecting cases of plagiarism of its marketing campaign on the web. The idea was to investigate to what extent this could be done using the plagiarism detection software which is available on the web. The poor results we obtained with WCopyFind tool, highlight the need to develop at-hoc plagiarism detection methods for SME's.

References

Barrón-Cedeño, A. and P. Rosso. 2009. On automatic plagiarism detection based on n-grams comparisons. *Proc. European Conference on Information Retrieval, ECIR-2009*, pages 696–700.

Barrón-Cedeño, A., P. Rosso, and J.M. Benedí. 2009. Reducing the plagiarism detection search space on the basis of the Kullback-Leibler Distance. *Proc. 10th Int. Conf. on Comput. Ling. and Intelligent Text Processing, CICLing-2009, Springer-Verlag, LNCS(5449)*, pages 523–534.

Dreher, H. 2007. Automatic conceptual analysis for plagiarism detection. *Journal of Issues in Informing Science and Information Technology 4*, pages 601–614.

Using Microsoft SQL Server platform for plagiarism detection

Vladislav Shcherbinin
American University of Nigeria
Lamido Zubairu way, Yola township by-pass,
PMB 2250, Yola, Nigeria
vladislav.scherbinin@gmail.com

Sergey Butakov
SolBridge International School of Business,
151-13 Samsung 1-Dong, Dong-gu, Daejeon,
300-814, South Korea
butakov@solbridge.ac.kr

Abstract: The paper presents an approach for plagiarism detection using Microsoft SQL Server platform in a large corpus of documents. The approach was used for participation in the first international plagiarism detection competition that was held as a part of PAN'09 workshop. The main advantages of the proposed approach are its high precision, good performance and readiness for deployment into a production environment with relatively low cost of the required third party software. The approach uses fingerprinting-based algorithm to compare documents and Levenstein's metric to markup plagiarized fragments in the texts.
Keywords: external plagiarism detection, Winnowing, document fingerprinting

1 Introduction

Digital plagiarism remains a burning issue both in academia and industry over the last two decades. Of course methods and tools of plagiarism uncovering have evolved a lot from the pioneering works on plagiarism uncovering in source codes in 1980s to web-enabled anti-plagiarism services of today.

Plagiarism detection methods at large can be split into two large groups: external document analysis methods and intrinsic plagiarism detection methods, or stylometry (Maurer, Kappe, & Zaka 2006). The method and software proposed in this paper aimed on the external plagiarism detection, e.g. revealing the text copied from other documents. The software was tested on the corpus of document provided for competition. The rest of the paper is organized as follows: the detailed description of the software platform and the detection process can be found in the second and third sections of the paper. Conclusion section summarizes the results and proposes directions for the future research.

2 Detection process

The document processing for the competition was performed by three nodes. Node 1 served as DBMS platform and Node 2 and Node 3 were used on the detection phase. The following subsections explain detection steps in details.

2.1 Loading and preprocessing of the documents

To perform the comparison on a large corpus of documents we decided to use the Winnowing, one of the well-known fingerprinting-based algorithms (Schleimer et al., 2003). According to this algorithm each document was substituted with a set of its hashes for the detection purposes.

The database designed to store documents and fingerprints consists of three tables: Folder, Document, and Fingerprint.

After loading documents and compiling their fingerprints the Fingerprint table was indexed with two indexes: one nonclustered index on hash value and document ID (index 1) and another clustered index on document ID, hash value and sequential number of a hash in the document (index 2). After the loading phase

Stein, Rosso, Stamatatos, Koppel, Agirre (Eds.): PAN'09, pp. 36-37, 2009.

the Fingerprint table was populated with 137,981,386 records. The most time consuming operation here was loading documents and compiling fingerprints.

2.2 Locating sources

The main objective of this step was to reduce the number of documents for comparison phase. This step selects all pairs of documents that share at least one fingerprint and stores these pairs in a table for more detailed analysis. After this step the table that links the pairs of possible matches in the documents was populated with only 44,532 records instead of 52,000,000 – possible number of pairs the search would have had to process if it compares all suspicious documents versus all source documents: 7214 * 7215 = 52,049,010. This step literally substituted the "one-vs-all" comparison with "one-vs-suspicions". As this step consists of only one query the better system performance could be achieved only by improving MS SQL Server hardware. This step uses index 1.

2.3 Detecting plagiarized passages

At this point all the required information is ready for the main step: detection of the common fragments in documents. The result from this step was used to identify exact plagiarized excerpts and to establish anchors for the further analysis. The main point here is the proper indexing of the Fingerprint table: on this step the clustered index created earlier (index 2) was used which provided the best possible execution plan.

After all common fingerprints have been identified and thus provided established anchors, the next task was to find common intervals for marking up the plagiarized passages. For better performance this process was distributed among two workstations (nodes 2 and 3), each running a console application performing the following steps:

1. Retrieve an unprocessed document from the Document table and corresponding records from the table that links it with possible sources.
2. For each record run the following steps:
    a. Execute the stored procedure to retrieve starting positions of the common excerpts.
    b. For each result skip forward character by character in both source and suspicious documents, while characters are equal. This will identify exact excerpt.
    c. Skip forward *n* characters, and compare excerpts using Levenstein's distance to identify near similar and obfuscated excerpts.
3. Save identified intervals into the DB.

Both nodes used several separate threads for this processing and each thread was processing a separate document, retrieved on the step 1 shown above. The detection time could be improved by increasing the computational power of the processing nodes (nodes 2 and 3) or by further increasing the number of nodes.

2.4 Compiling results

On the last step Microsoft SQL Server Integration Services was used to export information about detected plagiarism to XML files with the required format.

3 Conclusion

As the competition results indicate the proposed approach provides competitive results in terms of preciseness. Moreover it comes in the ready-to-deploy form that can be easily implemented on relatively inexpensive third party software (MS SQL Server). This will allow easy system integration with virtually any university-wide course management system. The required improvements to reduce the granularity of results are planned for implementation in the next version of the software. At this stage of the development the solution is publicly available for downloading as a desktop version at www.siberiasoft.info.

References

Maurer, H., Kappe F., Zaka B. (2006) Plagiarism – A Survey. Journal of Universal Computer Sciences, vol. 12, no. 8, pp. 1050 – 1084.

Schleimer S., Wilkerson D., and Aiken A. (2003). Winnowing: Local Algorithms for Document Fingerprinting. Proceedings of the ACM SIGMOD International Conference on Management of Data, pages 76-85, June 2003.

Intrinsic Plagiarism Detection Using Character *n*-gram Profiles

Efstathios Stamatatos
University of the Aegean
83200 - Karlovassi, Samos, Greece
stamatatos@aegean.gr

Abstract: The task of intrinsic plagiarism detection deals with cases where no reference corpus is available and it is exclusively based on stylistic changes or inconsistencies within a given document. In this paper a new method is presented that attempts to quantify the style variation within a document using character *n*-gram profiles and a style change function based on an appropriate dissimilarity measure originally proposed for author identification. In addition, we propose a set of heuristic rules that attempt to detect plagiarism–free documents and plagiarized passages, as well as to reduce the effect of irrelevant style changes within a document. The proposed approach is evaluated on the recently-available corpus of the 1st Int. Competition on Plagiarism Detection with promising results.
Keywords: Plagiarism detection, Character *n*-grams, Stylistic inconsistencies

1 Introduction

Textual plagiarism (the unacknowledged use of the work of another author either as an exact copy or a slightly modified version) is a major problem in modern world affecting education and research mainly. The rapid development of WWW made billions of web pages easily accessible to anyone providing plenty of potential sources for plagiarism. As a result, automated plagiarism analysis and detection receives increasing attention in both academia and software industry (Maurer et al, 2006).

There are two basic tasks in plagiarism analysis. In *external plagiarism detection* a reference corpus is given and the task is to identify pairs of identical or very similar passages from a suspicious document and some texts of the reference corpus. Most of the research studies in plagiarism analysis deal with this task (Hoad and Zobel, 2003; Stein, 2005). On the other hand, *intrinsic plagiarism detection* is more ambitious since no reference corpus is given (Meyer zu Eissen et al., 2007; Stein and Meyer zu Eissen, 2007). This task is applied in cases where it is not possible to have a representative reference corpus. In addition, the comparison of a suspicious document with all the texts of a very large corpus may be impractical in terms of computational time cost. It can also serve as a preprocessing step to an external plagiarism detection tool in order to reduce the time cost.

To handle the intrinsic plagiarism detection task one has to detect plagiarized passages of a suspicious document exclusively based on irregularities or inconsistencies within the document. Such inconsistencies or anomalies are mainly of stylistic nature.

The attempts to quantify writing style, a line of research known as ‘stylometry’, have a long history (Holmes, 1998). A great variety of measures that represent some kind of stylistic information have been proposed especially in the framework of authorship attribution research. In a recent survey, Stamatatos (2009) distinguishes the following types of stylometric features: lexical features (word frequencies, word *n*-grams, vocabulary richness, etc.), character features (character types, character *n*-grams), syntactic features (part-of-speech frequencies, types of phrases, etc.), semantic features (synonyms, semantic dependencies, etc.), and application-specific features (structural, content-specific, language-specific).

Although the lexical features are still the most popular, a number of independent recent studies have demonstrated the effectiveness of character *n*-grams for quantifying writing style (Keselj et al., 2003; Stamatatos, 2006;

Stein, Rosso, Stamatatos, Koppel, Agirre (Eds.): PAN'09, pp. 38-46, 2009.

Stamatatos, 2007; Kanaris and Stamatatos, 2007; Koppel et al., 2009). This type of features can be easily measured in any text and it is language and domain independent since it does not require any text pre-processing. These measures are also robust to noise. Note that in plagiarism analysis the efforts of an author to slightly modify a plagiarized passage may be considered as noise insertion. Graham et al. (2005) were the first to use character *n*-grams to detect stylistic inconsistencies in texts. However, their results were poor. One reason for this is that they only used character bigrams. Another reason is that the distance measure they used (cosine distance) was unreliable for very short texts. Note also that Graham et al. (2005) were based on predefined text segments (paragraphs) and their task was to identify whether two consecutive paragraphs differ in style or not.

In this paper, we propose a method for intrinsic plagiarism detection based on character *n*-gram profiles (the set of character *n*-gram normalized frequencies of a text) and an appropriate dissimilarity measure originally proposed for author identification. Our method automatically segments documents according to stylistic inconsistencies and decide whether or not a document is plagiarism-free. A set of heuristic rules is introduced that attempt to detect plagiarism on either the document level or the text passage level as well as to reduce the effect of irrelevant stylistic changes within a document.

The rest of the paper is organized as follows. Section 2 describes the method of quantifying stylistic changes within a document. Then, Section 3 includes the plagiarism detection heuristics while Section 4 describes the evaluation procedure. Finally, Section 5 discusses the main points of this study and proposes future work directions.

2 The style change function

The main idea of the proposed approach is to define a sliding window over the text length and compare the text in the window with the whole document. Thus, we get a function that quantifies the style changes within the document. Then, we can use the anomalies of that function to detect the plagiarized sections. In particular, the peaks of that function (corresponding to text sections of great dissimilarity with the whole document) indicate likely plagiarized sections. Therefore, what we need is a means to compare two texts knowing that one of the two (the text in the window) is shorter or much shorter than the other (the whole document).

Following the practice of recent successful methods in author identification (Keselj et al., 2003; Stamatatos, 2006; Stamatatos, 2007; Koppel et al., 2009; Stamatatos, 2009), each text is considered as a bag-of-character *n*-grams. That is, given a predefined *n* that denotes the length of strings, we build a vector of normalized frequencies (over text length) of all the character *n*-grams appearing at least once in the text. This vector is called the *profile* of the text. Note that the size of the profile depends on the text length (longer texts have bigger profiles). An important question is the value of *n*. A high *n* corresponds to long strings and better capture intra-word and inter-word information. On the other hand, a high *n* considerably increases the dimensionality of the profile. To keep dimensionality relatively low and based on preliminary experiments as well as on previous work on author identification (Stamatatos, 2007; Koppel et al., 2009) we used character 3-grams in this study. The complete set of parameter settings for the proposed method is given in Table 1. These settings were estimated using a small part (~200 documents) of the evaluation corpus (see section 4).

Description	Symbol	Value
Character *n*-gram length	n	3
Sliding window length	l	1,000
Sliding window step	s	200
Threshold of plagiarism-free criterion	t_1	0.02
Real window length threshold	t_2	1,500
Sensitivity of plagiarism detection	a	2

Table 1: Parameter settings used in this study.

Let $P(A)$ and $P(B)$ be the profiles of two texts A and B, respectively. Stamatatos (2007) studied the performance of various distance measures that quantify the similarity between two character *n*-gram profiles in the framework of author identification experiments. The following distance (or dissimilarity) measure has been found to be both accurate and robust when the two texts significantly differ in length.

$$d_1(A,B) = \sum_{g \in P(A)} \left(\frac{2(f_A(g) - f_B(g))}{f_A(g) + f_B(g)} \right)^2$$

where $f_A(g)$ and $f_B(g)$ are the frequency of occurrence (normalized over text length) of the *n*-gram *g* in text *A* and text *B*, respectively, Note that d_1 is not a symmetric function (typically, this means it cannot be called distance function). That is, only the *n*-grams of the first text are taken into account in the sum. This function is designed to handle cases where text *A* is shorter than text *B*. Stamatatos (2007) showed that d_1 is quite stable even when text *A* is much shorter than text *B*. This is exactly the case in the proposed method for intrinsic plagiarism detection where we want to compare a short text passage with the whole document that may be quite long. In this paper, we modified this measure as follows:

$$nd_1(A,B) = \frac{\sum_{g \in P(A)} \left(\frac{2(f_A(g) - f_B(g))}{f_A(g) + f_B(g)} \right)^2}{4|P(A)|}$$

where $|P(A)|$ is the size of the profile of text *A*. The denominator ensures that the values of dissimilarity function lie between 0 (highest similarity) and 1. We call this measure *normalized* d_1 (or nd_1).

Let *w* be a sliding window of length *l* (in characters) and step *s* (in characters). That is, each time the window is moved to the right by *s* characters and the profile of the next *l* characters is extracted. If *l*>*s* the windows are overlapping. Then, we can define the *style change* function (*sc*) of a document *D* as follows:

$$sc(i,D) = nd_1(w_i, D),\ i = 1 \ldots |w|$$

where $|w|$ is the total amount of windows (it depends on text-length). Given a text of *x* characters $|w|$ is computed as follows:

$$|w| = \left\lfloor 1 + \frac{x - l}{s} \right\rfloor$$

Examples of style change functions can be seen in figures 1, 2, 3, and 4.

3 Detecting plagiarism

3.1 Plagiarism on the document level

The first important question that must be answered is whether or not a given document contains any plagiarized passages. This is crucial to keep the precision of our method high. If we are unable to find documents that are plagiarism-free, it is quite likely for the plagiarism detection method to identify a number of text passages as the result of potential plagiarism for any given document. Thus, the credibility of the method would be very low.

There are two options to decide whether or not a document contains plagiarized sections:

By pre-processing: A criterion must be defined to indicate a plagiarism-free document. If this is the case, there is no further detection of plagiarized sections.

By post-processing: The algorithm detects any likely plagiarized sections and then a decision is taken based on these results.

Typically, the detected sections are compared to other sections of the document to decide whether there are significant differences between them (Stein and Meyer zu Eissen, 2007).

In this study we followed the former approach. The criterion we used is based on the variance of the style change function. If the document is written by one author, we expect the style change function to remain relatively stable. On the other hand, if there are plagiarized sections, the style change function will be characterized by peaks that significantly deviate from the average value. The existence of such peaks is indicated by the standard deviation. Let *S* denote the standard deviation of the style change function. If *S* is lower than a predefined threshold, then the document is considered plagiarism-free.

Plagiarism-free criterion: $S < t_1$

The value of the threshold t_1 was determined empirically at 0.02. Recall that the dissimilarity function we use is normalized. So, the definition of such a common threshold for all the documents is possible. However, the nd_1 measure is not independent of text length. Very short documents tend to have low style change function values. Moreover, very long texts are likely to contain stylistic changes made intentionally by the author. In both these cases this criterion will not be very accurate.

Figures 2 and 3 show the style change function of documents 00017 and 00034 of IPAT-DC (see section 4) that fall under the plagiarism-free criterion. The former is a successful case where no plagiarism exists. On the other hand, in the case of document 00034,

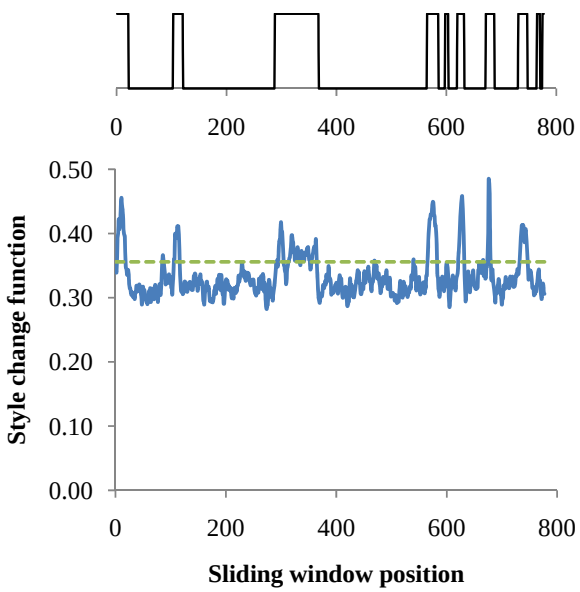

Figure 1: The style change function of document 00005 of IPAT-DC (solid line). The dashed line indicates the threshold of the plagiarized passage criterion. The binary function above indicates real plagiarized passages (high values).

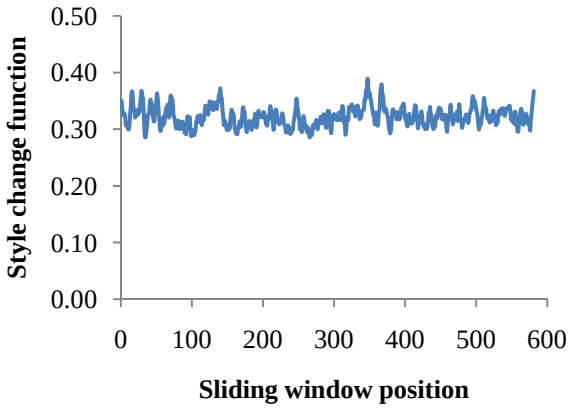

Figure 2: The style change function of document 00017 of IPAT-DC (a plagiarism-free document).

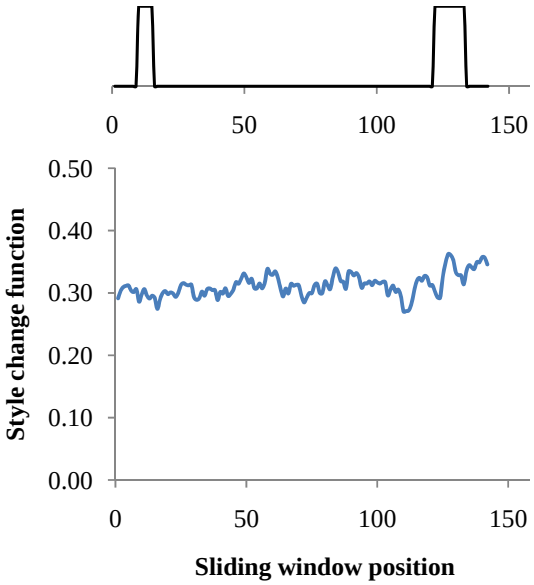

Figure 3: The style change function of document 00034 of IPAT-DC (false negative). The binary function above indicates real plagiarized passages (high values).

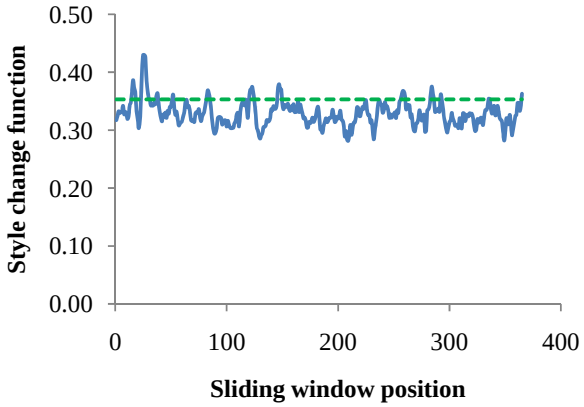

Figure 4: The style change function of the plagiarism-free document 00022 of IPAT-DC (a false positive). The dashed line indicates the threshold of the plagiarized passage criterion.

despite the presence of two plagiarized passages, the style change function fails to produce significant peaks that would increase its standard deviation. Note also that 00017 is longer than 00034 (more sliding windows in the *x*-axis) and the average style change function of 00017 is higher than that of 00034. Additionally, Figure 4 shows the style change of document 00022 of IPAT-DC. Although this document is plagiarism-free, the standard deviation of its style function is greater than the used threshold (false positive).

3.2 Identifying plagiarized passages

Given the style change function of a document, the task of plagiarism detection can be viewed as detecting peaks of that function corresponding to text sections that significantly differ from the rest of the document. One big

problem in plagiarism detection is that it is not possible to estimate the percentage of plagiarized text beforehand. In intrinsic plagiarism detection the problem is much harder since if the plagiarized sections are too long the stylistic anomalies would correspond to the style of the alleged author rather than the plagiarized sections. In this study we suppose that at least half of the text is not plagiarized so that the average of style change function would indicate the style of that author. However, the calculation of the average *sc* value would inevitably involve the plagiarized passages as well.

Let M and S denote the mean and standard deviation of *sc*, respectively. To reduce this problem we first remove from *sc* all the text windows with value greater than $M+S$. These text sections are highly likely to correspond to plagiarized sections. Let $sc(i',D)$ denote the style change function after the removal of these sections. Let M' and S' be the mean and standard deviation of $sc(i',D)$. Then, we define the following criterion to detect plagiarism:

Plagiarized passage criterion:
$sc(i',D) > M'+a*S'$

The parameter a determines the sensitivity of the plagiarism detection method. The higher the value of a, the less (and more likely plagiarized) sections are detected. The value of a was determined empirically at 2.0 to attain a good combination of precision and recall. Figures 1 and 4 show the result of applying the proposed criterion in two documents.

3.3 Detecting irrelevant style changes

An important factor that affects style using the character *n*-gram representation is the formatting of documents. A document written in uppercase with many space characters, punctuation symbols will have a quite different character *n*-gram profile than the same document in lowercase after the removal of any extra space and punctuation characters. The proposed method for the quantification of style changes is very general and is sensitive to such stylistic changes that are irrelevant to plagiarism. In fact, a very common technique to disguise plagiarism is to change the formatting of text. So, any plagiarism detection tool should attempt to reduce the formatting factor.

To deal with this problem, we performed a number of processes. First, each document is transformed to lowercase. Although the uppercase information is important for representing adequately the style of an author, it can be easily used to fool a plagiarism detection tool. Then, we removed from the profile of a text every character *n*-gram that contains no letter characters (a-z, or any lowercase character of foreign languages) at all. This way, any character *n*-gram that contains only digit, space, or punctuation characters, that is irrelevant to the content of text, is excluded and the formatting factor is reduced. Finally, the sliding window parameters operate on letter characters. That is, a window length of l characters means that the window should contain l letter characters. Note that all the other characters (digits, spaces, punctuation, etc.) are not removed. Therefore, if l=1,000, a window may contain 1,200 characters (this is the real window length) in total from which 1,000 are letter characters. Moreover, a step of s characters means that the window is moved to the right by s letter characters. This procedure ensures that all the text windows will have the same number of letter (or content) characters

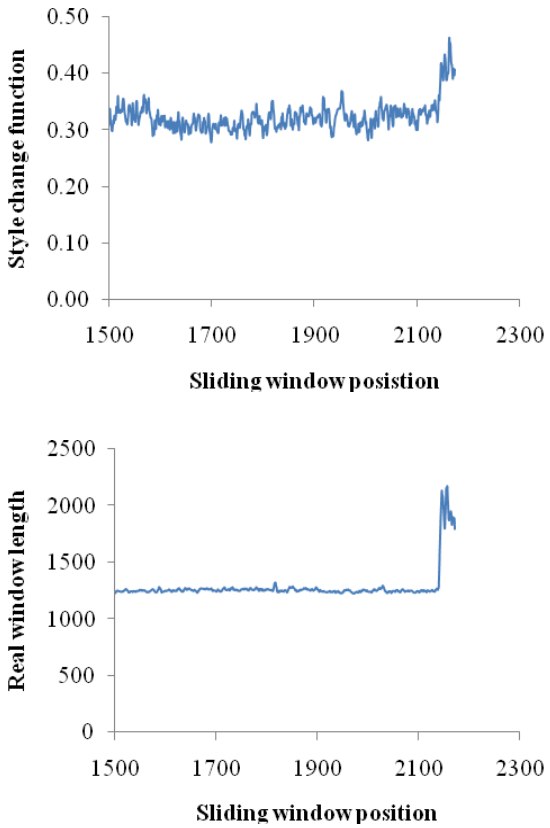

Figure 5: The style change function and the real window length of the last part of document 00046 of IPAT-DC.

and the formatting of the text will not significantly affect the style change function.

Since there is no prior knowledge on the genre of documents, a given document may be composed of several sections each one belonging to a different genre (or sub-genre) and therefore having different stylistic characteristics. For example, a table of contents has different style than the main document. The character *n*-gram representation is able to capture both the style of author and the style of genre but it is hard to distinguish these factors. To handle this problem, we make use of the real window length as defined above. In more detail, let l' be the real window length (the total number of characters included in a window that contains l letter characters) of a text section. The real window length is affected by some genres. For example, the l' of a table of contents is higher than the l' of the main document. This is demonstrated in figure 5 that shows the style change function and the real window length of the last part of document 00046 of IPAT-DC (for l=1,000). This document ends with an index. Note that the real window length of this special section is much higher than the rest of the document. The stylistic difference between the index and the rest of the document is captured by the style change function. However, this difference has nothing to do with plagiarism. To take such cases into account, an additional criterion was used to detect plagiarized passages:

Special section criterion: $l' < t_2$

This criterion is combined with the plagiarized passage criterion. Based on empirical evaluation, the value of the threshold t_2 was estimated at 1,500 (or 1.5l). Note that this criterion excludes text sections with overly real window length. However, one can take advantage of this criterion and disguise plagiarism by inserting many formatting characters to a text section so that l' is considerably increased. Moreover, a plagiarized section within a special section (e.g. table of contents) that resembles the style of that section will not be detected.

4 Evaluation

In the framework of the 1st International competition on plagiarism detection a large corpus has been released for the Intrinsic Plagiarism Analysis Task (Potthast et al., 2009). This corpus is segmented into a development part (IPAT-DC) and a competition part (IPAT-CC) each one comprising 3,091 documents. An artificial plagiarism tool has been used to automatically insert plagiarized passages within the documents. The following evaluation results are mainly based on IPAT-DC since this corpus also provides ground truth data. IPAT-DC comprises a wide variety of texts covering many genres and topics. The text length varies from (roughly) 3,000 characters to 2.5 million characters. Interestingly, the plagiarized passages begin in randomly selected positions covering arbitrary combinations of words, sentences, and paragraphs.

4.1 Evaluation on the document level

First, we evaluate the plagiarism-free criterion that operates on the document level. Table 2 shows the confusion matrix of IPAT-DC after the application of this criterion. It is important that over 70% of the plagiarism-free documents were correctly classified. This is crucial to keep the overall precision on reasonable level. On the other hand, false positives (see Figure 4) harm the precision while false negatives (see Figure 3) harm the recall.

	Actual	
Guess	Plagiarism-free	Plagiarized
Plagiarism-free	1102	545 (22%)
Plagiarized	443	1001 (78%)

Table 2: Confusion matrix (on the document level) after the application of the plagiarism-free criterion. The percentage of plagiarized passages included in the documents are inside parentheses.

As can be seen, roughly 1/3 of the plagiarized documents are considered plagiarism-free. However, taking into account the number of plagiarized passages within each document (indicated inside parentheses in the table), we see that 22% of the plagiarized passages is missed. So, the upper bound for the recall on the passage level will be 78%. A closer look to the false negatives shows that text-length is a crucial factor. Figure 6 depicts the distribution of false negatives over text-length of documents. As can be seen, the majority of false negatives are relatively short documents (<30K chars). Moreover, the shorter

a document, the more likely to be false negative.

Corpus	IPAT-DC	IPAT-CC
Recall	0.4552	0.4607
Precision	0.2183	0.2321
F-score	0.2876	0.3086
Granularity	1.22	1.25
Overall score	0.2358	0.2462

Table 3: Performance of the plagiarism passage criterion on two corpora (development and competition corpus).

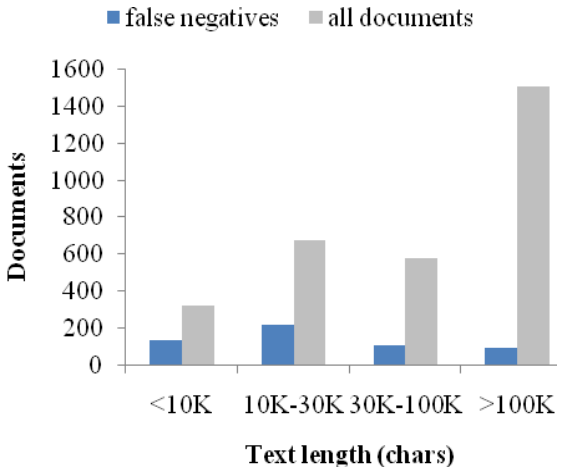

Figure 6: Distribution of false negatives over text length.

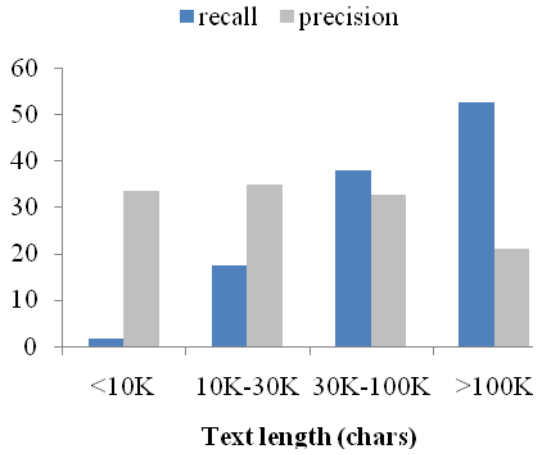

Figure 7: Recall and precision for varying text length.

4.2 Evaluation on the passage level

To evaluate the plagiarism detection method, we should first define appropriate measures. In particular, we used the performance measures defined in the framework of the 1st int. competition on plagiarism detection: recall, precision, granularity, and overall score. Let r denote a plagiarized passage and $|R|$ be the set of all plagiarized passages in the corpus. Moreover, let p be a detected passage by the proposed method, $|P|$ be the set of all detected passages, and $|R_p|$ be the subset of R that overlap with at least one member of $|P|$. Finally, let $|r|$ and $|\hat{r}|$ be the length of a plagiarized passage and the sum of its detected characters by the plagiarism detection method, respectively. Similarly, $|p|$ and $|\hat{p}|$ are the length of a detected passage and the sum of their chars that belong to any plagiarized passage. Then, recall, precision, and granularity can be defined as follows:

$$recall = \frac{1}{|R|}\sum_{i=1}^{|R|}\frac{|\hat{r}_i|}{|r_i|}$$

$$precision = \frac{1}{|P|}\sum_{i=1}^{|P|}\frac{|\hat{p}_i|}{|p_i|}$$

$$granularity = \log_2(1+\frac{1}{|R_P|}\sum_{i=1}^{|R_P|}|r_i \cap P|$$

$$overall = \frac{F}{granularity}$$

where $|r_i \cap P|$ denotes the number of different detections that overlap with the plagiarized passage r_i, and F is the harmonic mean of recall and precision. Essentially, the granularity measure indicates the fragmentation of the detected passages. A granularity value of 1 means that at most one detected section overlaps with a plagiarized passage.

The results of the evaluation of the plagiarized passage criterion are included in table 3 on the development and competition corpus of the intrinsic plagiarism analysis task (taken by the official results of the competition). The parameter values shown in table 1 have been used to produce these results. As can be seen, the performance of the proposed method remains stable for both corpora. Actually, the performance on IPAT-CC is better than on IPAT-DC that was used for estimating the values of parameters. This indicates that the proposed settings are quite general and robust.

Figure 7 provides a closer look in the recall-precision results on IPAT-DC with respect to text-length of documents. It is obvious that recall is dramatically affected by decreasing text length. The distribution of false negatives showed in figure 6 offers a reasonable explanation for this. Precision is more stable. However, it tends to decrease while text length increases.

5 Discussion

In this paper a new method for intrinsic plagiarism detection has been presented. The proposed approach is based on character *n*-gram profiles, a style change function using an appropriate dissimilarity measure as well as a set of heuristic rules to detect plagiarized passages. The evaluation results demonstrate that it is able to detect roughly half of the plagiarized sections. On the other hand, the precision remains low. An important factor for improving precision is the development of more sophisticated and accurate plagiarism-free criteria on the document level. The precision can also be improved by increasing the sensitivity parameter *a*. However, this will harm recall.

The proposed method is easy to follow and requires no language-dependent resources. Moreover, it requires no text segmentation or preprocessing. The proposed parameter settings proved to be effective when the approach was evaluated on the IPAT-CC. Note that the parameter values of table 1 were not optimized for IPAT-DC. However, the application of machine learning algorithms could improve the estimation of these parameters. Especially, the definition of the window length is crucial since it determines the shortest plagiarized passage that can be detected. On the other hand, a very short window would not adequately capture the stylistic properties of the text.

Another future work direction is to examine different schemes for comparing a text window with the whole document. The approach followed in this paper is fast since it requires the calculation of only one profile for the whole document. Alternative approaches include the comparison of the text window with the window complement (the document without the window) and the comparison of a text window with all the other text windows.

Finally, character *n*-grams of higher order could be used. Preliminary experiments using character 4-grams and 5-grams did not show significant improvement on the performance of the method. However, this remains to be carefully examined.

References

Graham, N. Hirst, G. and Marthi, B. (2005). Segmenting Documents by Stylistic Character. *Natural Language Engineering*, 11(4): 397-415.

Hoad, T.C. and J. Zobel. 2003. Methods for Identifying Versioned and Plagiarised Documents. *Journal of the American Society for Information Science and Technology*, 54(3):203–215.

Holmes, D.I. 1998. The Evolution of Stylometry in Humanities Scholarship. *Literary and Linguistic Computing*, 13(3): 111-117.

Kanaris, I. and E. Stamatatos. 2007. Webpage Genre Identification Using Variable-length Character *n*-grams, In *Proc. of the 19th IEEE Int. Conf. on Tools with Artificial Intelligence*, v.2, pp. 3-10.

Keselj, V., F. Peng, N. Cercone, and C. Thomas. 2003.. N-gram-based Author Profiles for Authorship Attribution. In *Proceedings of the Pacific Association for Computational Linguistics*, pp. 255-264.

Koppel, M., J. Schler, and S. Argamon. 2009. Computational Methods in Authorship Attribution, *Journal of the American Society for information Science and Technology*, 60(1): 9-26.

Maurer, H., F. Kappe, and B. Zaka. 2006. Plagiarism - A Survey. *Journal of Universal Computer Science*, 12(8): 1050-1084.

Meyer zu Eissen, S., B. Stein, and M. Kulig. 2007. Plagiarism Detection without Reference Collections. *Advances in Data Analysis*, pp. 359-366, Springer.

Potthast, M., A. Eiselt, B. Stein, A. Barron, and P. Rosso. 2009. Plagiarism Corpus PAN-PC-09. Webis at Bauhaus-Universitaet Weimar and NLEL at Universidad Polytecnica de Valencia. (http://www.webis.de/research/corpora)

Stamatatos, E. 2009. A Survey of Modern Authorship Attribution Methods, *Journal of*

the American Society for information Science and Technology, 60(3): 538-556.

Stamatatos, E. 2007. Author Identification Using Imbalanced and Limited Training Texts. In *Proceedings of the 4th International Workshop on Text-based Information Retrieval*, pp. 237-241.

Stamatatos, E. 2006. Ensemble-based Author Identification Using Character N-grams, In *Proc. of the 3rd Int. Workshop on Text-based Information Retrieval*, pp. 41-46.

Stein, B., and S. Meyer zu Eissen. 2007. Intrinsic Plagiarism Analysis with Meta Learning. In *Proceedings of the SIGIR Workshop on Plagiarism Analysis, Authorship Attribution, and Near-Duplicate Detection*, pp.45-50.

Stein, B. 2005. Fuzzy-Fingerprints for Text-Based Information Retrieval. In Proceedings of the 5th International Conference on Knowledge Management, J.UCS: 572–579.

External and Intrinsic Plagiarism Detection Using Vector Space Models

Mario Zechner, Markus Muhr, Roman Kern
Know-Center
8010 Graz
{mzechner, mmuhr, rkern}@know-center.at

Michael Granitzer
Know-Center Graz
Graz University of Technology
mgranitzer@tugraz.at

Abstract: Plagiarism detection can be divided in external and intrinsic methods. Naive external plagiarism analysis suffers from computationally demanding full nearest neighbor searches within a reference corpus. We present a conceptually simple space partitioning approach to achieve search times sub linear in the number of reference documents, trading precision for speed. We focus on full duplicate searches while achieving acceptable results in the near duplicate case. Intrinsic plagiarism analysis tries to find plagiarized passages within a document without any external knowledge. We use several topic independent stylometric features from which a vector space model for each sentence of a suspicious document is constructed. Plagiarized passages are detected by an outlier analysis relative to the document mean vector. Our system was created for the first PAN competition on plagiarism detection in 2009. The evaluation was performed on the challenge's development and competition corpora for which we report our results.
Keywords: Plagiarism Detection, Nearest Neighbor Search, Stylometry, Outlier Detection

1 Introduction

Plagiarism, defined as the theft of intellectual property (Maurer, Kappe, and Zaka, 2006), has been a problem for centuries not only in academic circles. There exist different forms of plagiarism, ranging from simply copying and pasting original passages to more elaborate paraphrased and translated plagiarism. Anecdotal evidence and studies such as (Sheard et al., 2002) strengthen the suspicion that plagiarism is on the rise, facilitated by new media such as the World Wide Web. Growing information sources ease plagiarism while plagiarism prevention and detection become harder.

To combat these problems the first competition on plagiarism detection was held at the third PAN workshop in 2009 in which we participated. The competition was split into two tasks: external plagiarism detection and intrinsic plagiarism detection. External plagiarism detection deals with the problem of finding plagiarized passages in a suspicious document based on a reference corpus. Intrinsic plagiarism detection does not use external knowledge and tries to identify discrepancies in style within a suspicious document. For both tasks extensive, machine generated training corpora were provided upon which we developed and evaluated our solutions. Our contribution is as follows:

- We present a robust external plagiarism detection method based on indexing by balanced clustering in a high dimensional term vector space with a focus on full and very near duplicate detection.
- We present an intrinsic plagiarism detection method based on a combined set of stylometric features spanning a vector space, using outlier analysis to determine plagiarized passages without a reference corpus or any other external knowledge source.
- We report our results for both problems on the PAN 09 development and competition corpora.

This paper is outlined as follows: in section 2 we present related work. Section 3 describes our system for the external plagiarism task, section 4 gives insight on our intrinsic plagiarism detection system. In section 5 we describe the PAN 09 dataset and report our results for the external and intrinsic plagiarism tasks. We conclude and give an outlook on future work in section 6

2 Related Work

External plagiarism detection relies on a reference corpus composed of documents from which passages might have been plagiarized.

Stein, Rosso, Stamatatos, Koppel, Agirre (Eds.): PAN'09, pp. 47-55, 2009.

A passage could be made up of paragraphs, a fixed size block of words, a block of sentences and so on. A suspicious document is checked for plagiarism by searching for passages that are duplicates or near duplicates of passages in documents within the reference corpus. An external plagiarism system then reports these findings to a human controller who decides whether the detected passages are plagiarized or not.

A naive solution to this problem is to compare each passage in a suspicious document to every passage of each document in the reference corpus. This is obviously prohibitive. The reference corpus has to be large in order to find as many plagiarized passages as possible. This fact directly translates to very high runtimes when using the naive approach.

External plagiarism detection is similar to textual information retrieval (IR) (Baeza-Yates and Ribeiro-Neto, 1999). Given a set of query terms an IR system returns a ranked set of documents from a corpus that best matches the query terms. The most common structure for answering such queries is an inverted index. An external plagiarism detection system using an inverted index indexes passages of the reference corpus' documents. For each passage in a suspicious document a query is send to the system and the returned ranked list of reference passages is analyzed. Such a system was presented in (Hoad and Zobel, 2003) for finding duplicate or near duplicate documents.

Another method for finding duplicates and near duplicates is based on hashing or fingerprinting. Such methods produce one or more fingerprints that describe the content of a document or passage. A suspicious document's passages are compared to the reference corpus based on their hashes or fingerprints. Duplicate and near duplicate passages are assumed to have similar fingerprints. One of the first systems for plagiarism detection using this schema was presented in (Brin, Davis, and Garcia-Molina, 1995).

External plagiarism detection can also be viewed as nearest neighbor problem in a vector space R^d. Passages are represented as vectors within this vector space. If passages from the reference corpus are "near enough" to a passage of the suspicious document in this vector space, the passage is marked as potentially plagiarized. The dimensions of the vector space are usually defined by features extracted from the passages such as terms (usually called the "bag of words" vector space model). This often yields high dimensional vector spaces. Neighbors are defined either by a distance metric $D : R^d \times R^d \rightarrow R$ or by a similarity function $S : R^d \times R^d \rightarrow R$. Smaller values for D signal nearness while high values for S signal similarity. Nearest neighbor searches in a vector space with a distance metric can be made sub linear by the use of space partitioning techniques that rely on the triangle inequality guaranteed by a metric. Famous partitioning schemes are the Kd-Tree (Bentley, 1975) or the metric tree (Ciaccia, Patella, and Zezula, 1997). However, these techniques degrade to linear searches for high dimensional vector spaces due to the curse of dimensionality: average inter point distances become more similar. This can be somewhat overcome by assuming that the data indeed lies on a lower dimensional manifold which can be captured by dimensionality reduction. In the reduced space partitioning schemas might be applicable again while the original neighborhoods are preserved by the reduction. Common dimensionality reduction approaches are Principle Component Analysis (M.E., 2003) for linear reduction or Isomap (Tenenbaum, de Silva, and Langford, 2000) for non linear reduction. These methods are generally very costly so other methods for nearest neighbor searches in high dimensional vector spaces have been devised. Locality sensitive hashing (LSH) (Gionis, Indyk, and Motwani, 1999) received a lot of attention in recent years due to its simplicity and theoretical guarantees. LSH is the equivalent of the aforementioned fingerprinting schemes applied to high dimensional vector spaces. The nearest neighbors returned by LSH are only approximate in nature. Another approximate nearest neighbor schema was introduced in (Chierichetti et al., 2007) that uses hierarchical cluster trees for space partitioning.

Intrinsic plagiarism detection only recently received attention from the scientific community. It was first introduced in (Meyer zu Eissen and Stein, 2006) and defined as detecting plagiarized passages in a suspicious document without a reference collection or any other external knowledge. A suspicious document is first decomposed into passages. For each passage a feature vector is constructed. Features are derived from sty-

lometric measures like the average sentence length or the average word length known from the field of authorship analysis. These features have to be topic independent so as to capture the style of an author and not the domain she writes about. Next a difference vector is constructed for each passage that captures the passages deviation from the document mean vector. Meyer zu Eissen and Stein (2006) assume that a ground truth is given, marking passages actually from the author of the suspicious document. A model is then trained based on one-class classification, using the ground truth as the training set. The model is then used to determine which passages are plagiarized. However, it is not clear how the ground truth is derived from a suspicious document when no information about the document is known beforehand.

3 External Plagiarism Detection

We treat external plagiarism detection as a nearest neighbor search in a high dimensional term vector space. This is motivated by the extensive literature that exists for the nearest neighbor search problem as well as its conceptual simplicity. Our system consists of three stages:

- Vectorization of the passages of each document in the reference corpus and partitioning of the reference corpus vector space.
- Vectorization of the passages of a suspicious document and finding each passage's nearest neighbor(s) in the reference corpus vector space. Detection of plagiarism for each suspicious document is based on its nearest neighbor list via similarity thresholding.
- Post processing of the detected plagiarized passages, merging subsequent plagiarized passages to a single block.

3.1 Reference Corpus Vectorization & Partitioning

We adopt the vector space model for textual data as given in (Salton, Wong, and Yang, 1975). Each unique term in the reference corpus is represented as a dimension in the vector space $\mathcal{R}^d$, where d is the number of unique terms. Instead of creating a single vector for a complete document we create vectors for each sentence in a document as we want to detect plagiarism on a per sentence level.

We use the OpenNLP Framework [1] for tokenization and sentence splitting. For each sentence in every reference corpus document a term frequency vector based on the sentence's lower cased tokens is constructed, excluding stop words based on a stop word list. We did not apply any stemming or lemmatization. The resulting vectors are then normalized to unit length to overcome difference in sentence length. We use the standard cosine similarity to asses the similarity between sentences given by:

$$\text{cosine similarity}(\mathbf{x}, \mathbf{y}) = \frac{\langle \mathbf{x}, \mathbf{y} \rangle}{\|\mathbf{x}\| \|\mathbf{y}\|}$$

were $\mathbf{x}, \mathbf{y} \in \mathcal{R}^d$. The denominator of the above equation can be dropped as all vectors are scaled to unit length.

To achieve sub linear nearest neighbor searches we implement a variation of cluster pruning (Chierichetti et al., 2007). The set of reference corpus sentence vectors is first clustered into l partitions using a balanced online spherical k-means implementation (Zhong, 2005). Balancing is crucial as it provides more equal runtimes when searching for nearest neighbors. The balancing is achieved by introducing a penality to clusters that have more samples than others during the clustering process. Each sentence vector is associated with a single partition, each partition has a representative centroid vector. Our approach deviates from cluster pruning as presented in (Chierichetti et al., 2007) by associating each sentence vector only with the nearest cluster. Additionally we store a sorted list of similarities for each cluster, holding the similarities between the centroid of the cluster and the sentence vectors associated with that cluster. The resulting structure serves as an index. It allows searching the approximate nearest sentences for a given query sentence.

3.2 Suspicious Document Vectorization & Plagiarism Detection

We vectorize a suspicious document in the same way we vectorize reference corpus documents. For each sentence vector of a suspi-

[1] http://opennlp.sourceforge.net/

cious document we search the reference corpus for the k most similar sentences as follows:

- Determine the nearest cluster to the query sentence based on the cosine similarity between the centroids and the query sentence.
- Find the position in the sorted similarity list the query sentence would be inserted at based on its similarity to the cluster centroid. Gather $\frac{k}{2}$ sentence vectors to the left and right of that position in the list.
- Perform the same search in the second nearest cluster returning $\frac{k}{2}$ potential nearest neighbors and merge the two sets of candidate reference corpus sentences.

We justify this procedure as follows: The cluster a vector belongs to is likely to also contain the vector's nearest neighbors. To increase the probability of catching most true nearest neighbors we also use the second nearest cluster. An original sentence in the reference corpus and a full duplicate in a suspicious document will belong to the same cluster as they have the same vector representation. Consequently both vectors have the same similarity with the centroid of that cluster. The search in a cluster's similarity list should thus return the duplicated sentence. This can fail if there are more than k vectors in the cluster having the same similarity with the centroid as the suspicious sentence vector. The outcome is dependent on the quality of the sentence splitter as this type of search for full duplicates relies on correct sentence boundaries. The schema will also work in case a sentence was plagiarized with small modifications, albeit with a much lower probability of finding the correct nearest neighbor. We thus expect our system to work well for detecting full duplicates, acceptable in the case of slightly modified sentences and poorly for highly obfuscated sentences. Figure 1 illustrates the function of the similarity list of a cluster.

With the presented schema we can reduce the search for the nearest neighbors of a sentence from being linear in the number of sentences in the reference corpus to roughly $O(l + k + k/2)$ cosine similarity evaluations per sentence, where l is the number of centroids or clusters, k is the number of vectors taken from the nearest cluster's similarity list and $\frac{k}{2}$ is the number of vectors taken from the second nearest cluster's similarity list.

A sentence of a suspicious document is marked as plagiarized if its cosine similarity to the most similar candidate sentence from the reference corpus exceeds a threshold α. This parameter allows controlling the sensitivity of the system. The outcome of this stage is a set of sentences from the suspicious document that are plagiarized with high probability. The information about the plagiarized sentences' position in the original documents is also retained.

3.3 Post Processing

The final stage assembles the sentences marked as plagiarized in the second stage to continuous blocks. This is accomplished by simply checking whether sentences marked as plagiarized are in sequence in the suspicious document. If this is the case they are merged. This is repeated until no more merging is possible. To further increase the recall, we compare a plagiarized sentence's left and right neighbor sentences to the neighbors of the original sentence. These might have been missed in the nearest neighbor search and have now a chance to be detected. The neighborhood sentences are again compared via the cosine similarity and marked as plagiarized if the similarity to an original sentence is above some threshold β.

4 Intrinsic Plagiarism Detection

Our intrinsic plagiarism detection system is based on the ideas presented in (Meyer zu Eissen and Stein, 2006) and (Grieve, 2007). Meyer zu Eissen and Stein were the first to define the problem of intrinsic plagiarism detection: determine whether passages in a suspicious document are plagiarized based only on changes in style within the document. An author's style is also of importance in the field of authorship classification. Both problems rely on so called stylometric features. These features should be topic and genre independent and reflect an author's style of writing. Changes of style within a document can be detected by various methods. We choose a simple outlier detection scheme based on a vector space spanned by various stylometric features. The system is composed of 3 stages:

- Vectorization of each sentence in the suspicious document.

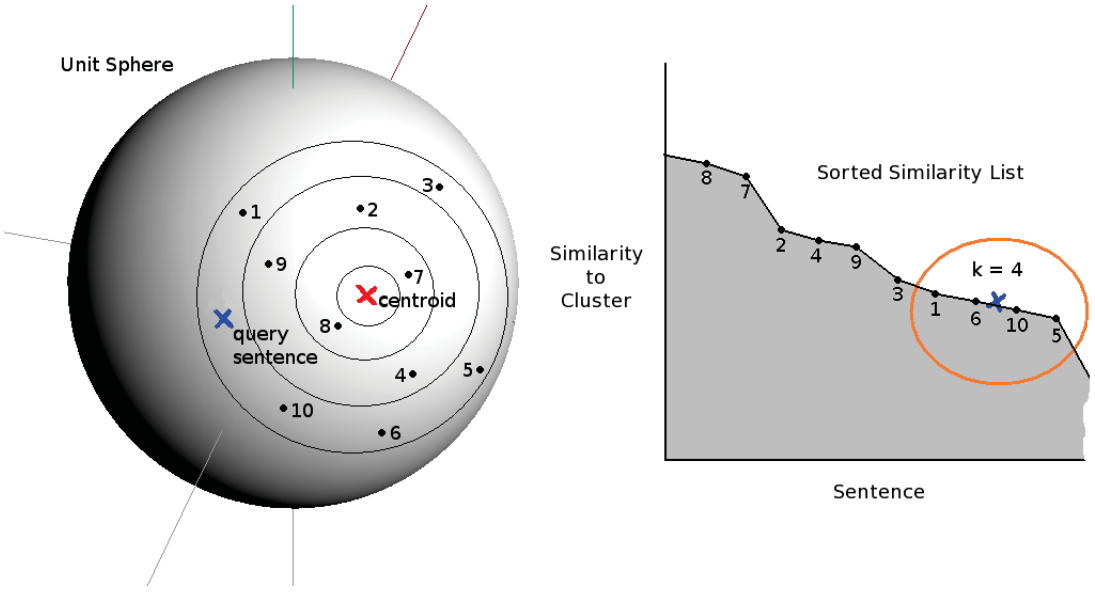

Figure 1: A cluster with its centroid, ten associated sentence vectors as well as a query sentence vector. The similarity sorting induces concentric rings of near equal similarity around the centroid. The search of the query sentence in the sorted similarity list with $k = 4$ is illustrated to the left.

- Determination of outlier sentences based on the document's mean vector.
- Post processing of the detected outlier sentences.

4.1 Suspicious Document Vectorization

Plagiarism is determined on a per sentence level in our intrinsic plagiarism detection system. However, we chose to use a window of k sentences around a given suspicious sentence. The window is composed of $\frac{k}{2}$ sentences to the left and right of the sentence. We adopt various stylometric features as presented in (Meyer zu Eissen and Stein, 2006) and (Grieve, 2007) that together form a single vector space:

- **Average word frequency class:** Each token w in a sentence window is assigned to an average word frequency class. The class is calculated by $\lfloor \log(freq_{w*}/freq_w \rfloor$, where $freq_{w*}$ is the absolute number of occurances of the most frequent word in a huge corpus and $freq_w$ is the number of occurances of the token in that same corpus. We derrived our frequency table from tokenizing the English Wikipedia, resulting in aproximately 6 million unique terms. Each average word frequency class is represented by a single dimension in the final vector space. The values in these dimension specify the number of tokens belonging to that class.
- **Punctuation:** For each sentence window the number of occurances of a certain punctuation character is measured. Each punctuation character is represented by a single dimension in the final vector space. The values in these dimensions reflect the frequencies of punctuation characters in the sentence window. [2]
- **Part of speech tags:** Each token w in a sentence window is assigned a part of speech tag from the Penn Treebank part of speech tag set. Each part of speech tag is represented by a dimension in the final vector space. The values in these dimensions reflect the frequencies of part of speech tags in the sentence window.
- **Pronouns:** For each sentence window the number of occurances of a certain pronoun is measured. Each pronoun is represented as a single dimension in the final vector space. The values reflect the

[2] We used the following punctuation characters in our experiments: .,?!:;()-

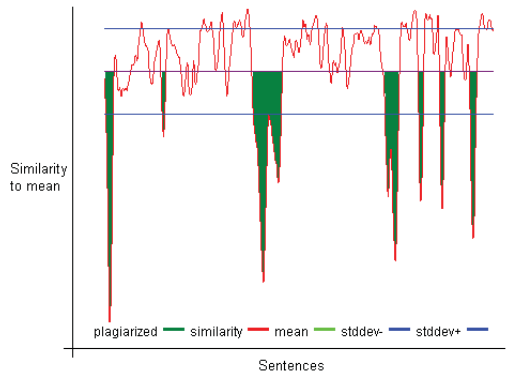

Figure 2: Similarity curve for a document. The x-axis shows the sentence as they appear in the document, the y-axis shows the similarity of the sentence with the document's mean vector. Filled areas indicate the location of actually plagiarized sentences.

frequencies with which each pronoun occured in a sentence window. [3]

- **Closed class words:** For each sentence window the number of occurances of a certain stop word is measured. Each stop word is represented by a single dimension in the final vector space. The values reflect the frequencies of stop words in a sentence window. We used the stop word list from the Snowball stemming framework [4].

For each sentence we construct a vector for each stylometric feature space based on the sentence's window. Each vector is normalized to unit length. The vectors of a sentence are then combined to a single vector by concatenation. The resulting vector is again normalized. Based on the final vectors of all sentences we construct a document mean vector.

4.2 Outlier Detection

The outlier detection tries to determine which sentences deviate from the document's mean. We use a simple detection scheme: we measure the cosine similarity from the mean vector to each sentence vector. We smooth the list of similarities by an average window smoothing procedure, where the size of the window l is a parameter. We determine the mean cosine similarity as well as the standard deviation from this list of similarities as:

$$mean = \frac{1}{n}\sum_{i=1}^{n} cos(\mathbf{v_i}, \mathbf{m})$$

$$stddev = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(cos(\mathbf{v_i}, \mathbf{m}) - mean)^2}$$

Where n is the number of sentences, $\mathbf{v_i}$ is the i^{th} sentence vector in the document, $\mathbf{m}$ is the document mean vector and cos is the cosine similarity. The j^th sentence is marked as an outlier if the following inequality holds:

$$cos(\mathbf{v_j}, \mathbf{m}) < mean - \epsilon * stddev$$

where ϵ is some small constant ≥ 1, and $\mathbf{v_j}$ is the sentence's vector. Marked sentences form the input to the last stage of the system. Figure 2 presents the similarity curve obtained for the suspicious document 5 in the development corpus of the PAN 09 challenge after smoothing.

4.3 Post Processing

Based on the sentences that deviate to much from the mean we derive the final blocks of plagiarized passages. As in the case of external plagiarism detection we simply merge all sentences marked that are neighbors until no further merging is possible.

5 Evaluation

We evaluated our system on the development corpora of the PAN 09 plagiarism detection competition. We describe description of the dataset as well as precision, recall, granularity and f1-measure results achieved by our system for various parameter settings. The measures are defined as follows:

$$precision = \frac{1}{|S|}\sum_{i=1}^{|S|}\frac{\#\text{detected chars of } s_i}{|s_i|}$$

$$recall = \frac{1}{|R|}\sum_{i=1}^{|R|}\frac{\#\text{plagiarized chars of } r_i}{|r_i|}$$

[3] We used the following pronouns: i, you, he, she, it, we, you, they, me, you, him, her, it, us, you, them, myself, yourself, himself, herself, itself, ourselves, yourselves, themselves, mine, yours, his, hers, its, ours, yours, theirs, my, your, his, her, its, our, your, their.

[4] http://snowball.tartarus.org/

$$f1 = \frac{2 * precision * recall}{precision + recall}$$

$$granularity = log_2(1 + \frac{1}{|S_R|} \sum_{i=1}^{|S_R|} \#\text{detections of} s_i \text{in} R)$$

were S is the set of detected passages and s_i is a single detected passage, R is the set of real plagiarized passages and r_i is a single plagiarized passage and S_R is the set of real plagiarized passages for which at least one detection exists. $|S|$, $|R|$ and $|S_R|$ denotes the number of passages in a set, $|s_i|$ and $|r_i|$ denote the number of characters in a passage.

5.1 PAN 09 Development Corpora

For each of the two tasks of the PAN 09 competition a development corpus was available.

The external plagiarism detection dataset consisted of 7214 suspicious documents and as many reference documents. Artificial plagiarized passages were added to the suspicious documents via an artificial plagiarist. For each document the plagiarist decided whether or not to plagiarize, from which reference documents to plagiarize, how many passages to plagiarize, which type of plagiarism to use for each passage and how long each passage would be. The type of plagiarism could either be obfuscated or translated plagiarism. Obfuscation was achieved by shuffling and deleting words, inserting words from an external source and replacing words with synonyms, antonyms, hypernyms or hyponyms. Obfuscation levels ranged from none to low to high.

For the intrinsic plagiarism detection task a corpus of 3091 suspicious documents was available. Plagiarized passages were generated similar to the external task.

5.2 External Plagiarism Detection Results

We performed a parameter study, evaluating precision, recall and f1-measure, for 500 randomly drawn suspicious documents from the external plagiarism development corpus. We studies the effect of the following parameters:

- l, the number of clusters for the index.
- k, the number of candidates taken from the similarity lists.
- α, the threshold above which the similarity between a suspicious and reference sentence has to be so that the suspicious sentence is marked as plagiarized.
- β, the threshold above which the similarity between neighbors of a marked sentence and the neighbors of the original sentence have to be in order to be marked as plagiarized.

Table 5.2 gives the results for various parameter settings. The threshold α was set dynamically depending on the length of a sentence. Very small sentences of five words, which are most likely sentence splitting errors or headlines, are completely ignored. For smaller sentences with fewer than fifteen words, the threshold is set to to 0.7. Sentences with up to thirty five words the threshold is set to 0.5, for longer sentences the threshold is set to 0.4. The threshold β for expanding detected blocks is set to 0.4 with the reasoning that it is very unlikely that nearby sentences will have high similarity values by chance without being actually copied. We arrived at this settings via manual trial and error on a small subset of suspicious documents.

The results show that the influence of the number of centroids l as well as the number of candidates k is marginal. In fact only a considerable change of the parameters to $l = 500$ and $k = 2000$ can affect the recall significantly. However, using such high settings degrades the nearest neighbor search for a sentence according to the previously stated complexity of $O(l + k + \frac{k}{2})$ similarity measurements. We believe that the marginal improvement of the precision and recall due to higher values of l and k do not compensate the much higher runtime. For larger corpora l and k would have to be set even higher, further increasing the runtime. We thus suggest using moderate values for l and k and instead focus on tuning α and β. Especially β is a good candidate to improve the overall recall.

5.3 Intrinsic Plagiarism Detection Results

We evaluated precision, recall and f1-measure for all suspicious documents of the internal plagiarism development corpus for various parameter settings. The parameters of the system are as follows:

- k, the size of the sentence window
- l, the size of the smoothing window by which the similarity list is smoothed.

l - k	*Precision*	*Recall*	*F1-Measure*	*Granularity*	*Recall None*	*Recall Low*
50 - 2	0.9616	0.4045	0.5695	1.9817	0.7044	0.4937
50 - 20	0.9523	0.4119	0.5750	1.9774	0.7053	0.4983
50 - 200	0.9411	0.4210	0.5818	1.9738	0.7053	0.5075
100 - 2	0.9597	0.4101	0.5746	1.9767	0.7044	0.4954
200 - 2	0.9419	0.4132	0.5745	1.9739	0.7050	0.4988
500 - 2000	0.8149	0.4782	0.6027	1.8497	0.7027	0.5534
Competition	0.6051	0.3714	0.4603	2.4424	-	-

Table 1: Results for the extrinsic plagiarism detection system on a split of the development corpus. The split contains 500 plagiarized documents plus the original documents from which the plagiarized passages were taken. The last row shows the results on the competition corpus. The postprocessing was always the same. The first column contains the number of centroids l and the number of evaluated neighbors k in the similarity list. Each row holds the precision, recall, f1-measure and granularity for all obfuscation levels. The column *Recall None* presents the recall on none obfuscated plagiarized passages and *Recall Low* contains the recall on none and low obfuscated plagiarized passages. The discrepancies between the measure on the development corpus split and the competition corpus are due to β being to low for the competition corpus.

Feature Space (k-l)	*Precision*	*Recall*	*F1-Measure*	*Granularity*
Word Freq. Class (6-3)	0.2215	0.0934	0.1314	-
Punctuation (12-9)	0.1675	0.1908	0.1784	-
Part of Speech Tags (6-6)	0.1797	0.1791	0.1794	-
Pronouns (12-9)	0.1370	0.3587	0.1983	-
Closed Class Words (12-9)	0.1192	0.1467	0.1316	-
Combined Feature Space (12-6)	**0.1827**	**0.2637**	**0.2159**	-
Competition Corpus	0.1968	0.2724	0.2286	1.2942

Table 2: Results for the intrinsic plagiarism detection system on the development corpus. Each cell holds the precision, recall and f1-measure for a given feature space and parameter setting. We only present the top performing parameter settings due to space constraints. We did not evaluate the granularity on the development corpus.

- ϵ, the constant by which the standard deviation of the similarity list is multiplied

Table 5.3 gives the results for various feature spaces and settings for the parameters k and l. Parameter ϵ was fixed at 1.1 for all the experiments. We varied k from 6 to 12 in steps of 2 and l from 3 to 12 in steps of 3. The ranges arose from preliminary experiments where they gave the best results.

We performed the experiment for each separate feature space in order to determine which features have high discriminative power. Surprisingly, pronouns performed very well when compared to more sophisticated features like the average word frequency class. It should be noted that pronouns do not fulfill the constraint of genre independence. A play by Shakespeare is likely to contain many more pronouns then a manual. We can also reproduce the results by Grieve (2007) for the punctuation feature which performed acceptable as well.

Based on the results for the separate feature spaces we took the three best performing spaces, part of speech tags, pronouns and punctuation as the basis for the combined vector space. This combined vector space was then again evaluated on the complete development corpus with varying parameters, showing significant improvements in all measures compared to the separate feature spaces. We used the best parameter settings found in this evaluation for the competition corpus.

6 Conclusion and Future Work

We presented our methods and results for the intrinsic and external plagiarism task of the PAN 09 competition. Our systems performed acceptable, taking the 5th out of 13 places in

the external task, the 3rd out of 4 places in the intrinsic task and the 5th overall place place out of 13 in the competition.

We plan on extending our external plagiarism detection system by incorporating term expansion via synonyms, hyponyms and hypernyms in order to cope with highly obfuscated plagiarism. We also plan to use a more sophisticated cluster pruning scheme that is hierarchical in nature, using hebbian learning to construct a topology of the search space to further increase the probability that the true nearest neighbor of a vector can be determined.

For future work for the intrinsic plagiarism problem, we aim at a better outlier detection method. We will also try to analyze and incorporate more stylometric features, combining them with the best performing features found in this competition. Dynamically adapting the parameters k and l for each document as well as for each feature space is also planed.

References

Baeza-Yates, Ricardo and Berthier Ribeiro-Neto. 1999. *Modern Information Retrieval*. Addison Wesley, May.

Bentley, Jon Louis. 1975. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517.

Brin, S., J. Davis, and H. Garcia-Molina. 1995. Copy detection mechanisms for digital documents. In *ACM International Conference on Management of Data (SIGMOD 1995)*.

Chierichetti, Flavio, Alessandro Panconesi, Prabhakar Raghavan, Mauro Sozio, Alessandro Tiberi, and Eli Upfal. 2007. Finding near neighbors through cluster pruning. In *PODS '07: Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 103–112, New York, NY, USA. ACM.

Ciaccia, Paolo, Marco Patella, and Pavel Zezula. 1997. M-tree: An efficient access method for similarity search in metric spaces. In Matthias Jarke, Michael J. Carey, Klaus R. Dittrich, Frederick H. Lochovsky, Pericles Loucopoulos, and Manfred A. Jeusfeld, editors, *VLDB*, pages 426–435. Morgan Kaufmann.

Gionis, Aristides, Piotr Indyk, and Rajeev Motwani. 1999. Similarity search in high dimensions via hashing. In *VLDB '99: Proceedings of the 25th International Conference on Very Large Data Bases*, pages 518–529, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Grieve, Jack. 2007. Quantitative authorship attribution: An evaluation of techniques. *Lit Linguist Computing*, 22(3):251–270, September.

Hoad, Timothy C. and Justin Zobel. 2003. Methods for identifying versioned and plagiarized documents. *J. Am. Soc. Inf. Sci. Technol.*, 54(3):203–215.

Maurer, Hermann, Frank Kappe, and Bilal Zaka. 2006. Plagiarism - a survey. *Journal of Universal Computer Science*, 12(8):1050–1084.

M.E., Timmerman. 2003. Principal component analysis (2nd ed.). i. t. jolliffe. *Journal of the American Statistical Association*, 98:1082–1083, January.

Meyer zu Eissen, Sven and Benno Stein. 2006. Intrinsic plagiarism detection. In Mounia Lalmas, Andy MacFarlane, Stefan M. Rüger, Anastasios Tombros, Theodora Tsikrika, and Alexei Yavlinsky, editors, *ECIR*, volume 3936 of *Lecture Notes in Computer Science*, pages 565–569. Springer.

Salton, G., A. Wong, and C. S. Yang. 1975. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620.

Sheard, Judy, Martin Dick, Selby Markham, Ian Macdonald, and Meaghan Walsh. 2002. Cheating and plagiarism: perceptions and practices of first year it students. *SIGCSE Bull.*, 34(3):183–187.

Tenenbaum, J. B., V. de Silva, and J. C. Langford. 2000. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, December.

Zhong, Shi. 2005. Efficient online spherical k-means clustering. In *Neural Networks, 2005. IJCNN '05. Proceedings. 2005 IEEE International Joint Conference on*, volume 5, pages 3180–3185 vol. 5, July-4 Aug.

Intrinsic Plagiarism Detection using Complexity Analysis

Leanne Seaward and Stan Matwin
University of Ottawa
2096 Madrid Avenue, Ottawa, ON,K2J 0K4
leanne_seaward@yahoo.ca, stan@site.uottawa.ca

Abstract: We introduce Kolmogorov Complexity measures as a way of extracting structural information from texts for Intrinsic Plagiarism Detection. Kolmogorov complexity measures have been used as features in a variety of machine learning tasks including image recognition, radar signal classification, EEG classification, DNA analysis, speech recognition and some text classification tasks (Chi and Kong, 1998; Zhang, Hu, and Jin, 2003; Bhattacharya, 2000; Menconi, Benci, and Buiatti, 2008; Frank, Chui, and Witten, 2000; Dalkilic et al., 2006; Seaward and Saxton, 2007; Seaward, Inkpen, and Nayak, 2008). Intrinsic Plagiarism detection uses no external corpus for document comparison and thus plagiarism must be detected solely on the basis of style shifts within the text to be analyzed. Given the small amount of text to be analyzed, feature extraction is of particular importance. We give a theoretical background as to why complexity measures are meaningful and we introduce some experimental results on the PAN'09 Intrinsic Plagiarism Corpus. We show complexity features based on the Lempel-Ziv compression algorithm slightly increase performance over features based on normalized counts. Furthermore we believe that more sophisticated compression algorithms which are suited to compressing the English language show great promise for feature extraction for various text classification problems.
Keywords: plagiarism, Kolmogorov, complexity, compression, machine learning

1 Introduction

Intrinsic plagiarism analysis involves analyzing a document for style changes which would suggest that certain passages have been written by a different author and are therefore plagiarized. It is closely related to authorship attribution and stylometry (Stamatatos, Fakotakis, and Kokkinakis, 2000; Stein and Meyer zu Eissen, 2007). Intrinsic plagiarism analysis is a very challenging problem because one has a small amount of text for global analysis and one must locally analyse very small portions or chunks of that text for style shifts. Authorship attribution normally uses several documents for author fingerprinting and tests possible authorship on an entire text document.

Because of the limited data available for this task and the difficulty of the problem, feature extraction is very important. Plagiarism analysis tools and authorship attribution models attempt to fingerprint an author's individual writing style using style features such as normalized counts of lexical and vocabulary richness features such as nouns, verbs, stop words, syllables per word etc (Stamatatos, Fakotakis, and Kokkinakis, 2000; Stein and Meyer zu Eissen, 2007). In addition one may analyze a document for topic or cohesion words. One may also use readability indexes to determine if the level of writing shifts (Stein and Meyer zu Eissen, 2007).

Features are extracted globally (for the entire document) and then locally (per sentence or paragraph chunk). With the exception of n-gram methods, the text is generally viewed as a bag-of-words and structure is ignored. We introduce a method of using compression to extract Kolmogorov complexity features which contain information about the structure of style features within the text. Extracting such features is scalable and complexity features can be used in state-of-the-art machine learning algorithms such as Support Vector Machines, Neural Networks and Bayesian Classifiers. The small text sample makes complexity analysis more difficult than for the authorship attribution problem. However, this method still shows promise and given the difficulty of the problem, a modest improvement is still important.

2 Introduction to Plagiarism Detection

There are two main types of plagiarism analysis - Intrinsic and Extrinsic. Extrinsic pla-

giarism analysis compares the document of interest to a corpus of reference documents (web pages, text books etc.) and tries to find passages which were copied from the reference collection. In contrast, intrinsic plagiarism detection uses no reference collection and tries to determine plagiarized passages by analyzing style changes within the document. Intrinsic plagiarism detection is closely related to author fingerprinting or stylometry.

Most research in plagiarism analysis focuses on extrinsic plagiarism analysis. If one assumes that the reference collection is complete, then extrinsic plagiarism analysis is a somewhat easier problem due to the fact that one must simply find the match between the plagiarized passage and the corresponding passage in the reference collection. The difficulty lies in reducing the computation time and detecting obfuscation attempts.

Obtaining a reference collection of all possible sources of plagiarism is impossible. Not all books are in electronic format and indexing all books for inclusion in such a corpus is a formidable task. There is always the possibility that a student has plagiarized from a document which is not available for indexing such as a paper from another student at another university.

One imagines that a robust plagiarism analysis tool would use both intrinsic and extrinsic plagiarism analysis. This is similar to the way a human expert such as a teacher or professor would analyze student papers for plagiarism. One may also use intrinsic plagiarism analysis to pre-select suspicious passages which can then be passed to an extrinsic plagiarism detector. It is always more desirable to have access to the plagiarized document as this removes all doubt as to the suspected plagiarism.

Intrinsic plagiarism analysis is related to authorship attribution and generally uses stylometry features which may consist of normalized counts of lexical features such as nouns and verbs as well as measures such as average sentence length and average word length. Intrinsic plagiarism detection may also use readability indexes and as measures which compute the divergence of the distribution of lexical elements to the expected probability distribution. With the exception of readability indexes, features are extracted as if each chunk in the text is a bag-of-words. Humans do not read or write Bags-of-words and so this approach is counterintuitive and loses information.

The need arises for a way of measuring structure of a text in a meaningful way which can be used as a feature in style analysis. It is also necessary that such a measure can be computed in an efficient and scalable manner.

The structure which we are measuring must be meaningful for the classification task at hand. We propose Kolmogorov Complexity measures as a way of measuring structural complexity of lexical elements in order to fingerprint author style.

3 Kolmogorov Complexity Measures

This paper introduces Kolmogorov complexity measures as style features in intrinsic plagiarism analysis. The basic idea is that each segment of text has a distribution with respect to a set of word classes. For example with respect to the word class noun – the text has a distribution of noun words and non-noun words. This can be though of as a binary string which has a 1 for each noun word and a 0 for each non-noun word. This binary string represents the distribution of noun words in the text.

For example, suppose we have the string: “Billy walked the dog yesterday.” The nouns are “Billy” and “dog”, the noun distribution is ’10010’. Likewise the only verb is “walked” so the verb distribution is ’01000’. Similarly if we look at short words (those with one syllable) vs. long words the distribution is ’11001’. There is a different distribution for any possible class of word type.

In Figure 1 we see how a text can be decomposed into a representation for each word class. Once we have this decomposition we would then like to quantify the structure for use in a machine learning algorithms.

Two sentences may have the same ratio for a particular feature but the distribution could be different. Suppose two sentences have the following structure for short words vs. long words.

0100001111010000100011110100000010000000111111000000111100000001111

Both representations have the same number of long vs. short words (0 vs. 1) but the

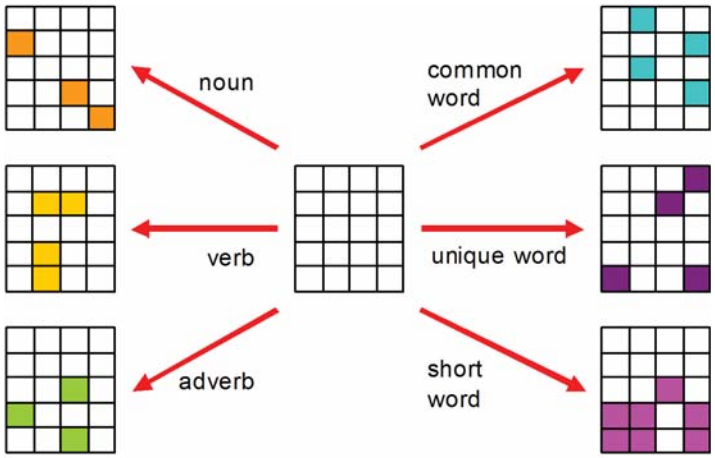

Figure 1: Decomposing a sentence into a variety of word class distributions for complexity analysis

first representation is more random and complex than the second. It is desirable to quantify this degree of randomness or complexity. One such method of doing so is Kolmogorov complexity measures.

4 Kolmogorov Complexity

Kolmogorov complexity, also known as algorithmic entropy, stochastic complexity, descriptive complexity, Kolmogorov-Chaitin complexity and program-size complexity, is used to describe the complexity or degree of randomness of a binary string. It was independently developed by Andrey N. Kolmogorov, Ray Solomonoff and Gregory Chaitin in the late 1960's (Li and Vitanyi, 1997).

In computer science, all objects can be viewed as binary strings. Thus we will refer to objects and strings interchangeably in this discussion. The Kolmogorov complexity of a binary string is the length of the shortest program which can output the string on a universal Turing machine and then stop (Li and Vitanyi, 1997).

It is impossible to compute the Kolmogorov complexity of a binary string. However there have been methods developed to approximate it. The Kolmogorov complexity of a string x, denoted as $K(x)$, can be approximated using any lossless compression algorithm (Li and Vitanyi, 1997). A compression algorithm is one which transforms a string A, to another shorter string, B. The associated decompression algorithm transforms B back into A or a string very close to A. A lossless compression algorithm is one in which the decompression algorithm exactly computes A from B and a lossy compression algorithm is one in which A can be approximated given B. When Kolmogorov Complexity, or $K(x)$, is approximated, this approximation corresponds to an upper-bound of $K(x)$ (Li and Vitanyi, 1997). Let C be any compression algorithm and let $C(x)$ be the results of compressing x using C. The approximate Kolmogorov complexity of x, using C as a compression algorithm, denoted $K_c(x)$, can be defined as follows:

$$K_c(x) = \frac{Length(C(x))}{Length(x)} + q$$

where q is the length in bits of the program which implements C. In practice, q is usually ignored as it is not useful in comparing complexity approximations and it varies according to which programming language implements C. If C was able to compress x a great deal then $K_c(x)$ is low and thus x has low complexity. Likewise if C could not compress x very much then $K_c(x)$ is high and x has high complexity.

5 Compression Algorithms and Kolmogorov Complexity Analysis

Kolmogorov complexity can be computed using any lossless compression algorithm. Once the text to be analyzed has been converted into a binary form related to a particular word class distribution, one simply applies a

compression algorithm to determine the degree to which it is compressed. If it compresses a great deal then complexity is high and vice versa.

Our previous research has used generic compression algorithms such as run-length encoding and Lempel-Ziv (Zlib) compression. For this intrinsic plagiarism detection task, Zlib compression was used. It may be especially interesting to investigate compression algorithms which assume prior knowledge about the probabilities of lexical features or which are designed to maximize compression for language texts in a particular language.

Frank et al. (2000) investigated text categorization using statistical data compression techniques. They use a corpus of two classes of documents and train a statistical compression tool (prediction by partial matching or PPM) using each corpus. They attempt to classify documents by determining which compression model compresses it the most. They conclude that data compression techniques perform well but are inferior to state of the art machine learning techniques such as SVM or Neural Nets. No attempt was made to merge compression features with machine learning algorithms.

Thus we have three possibilities for compression algorithms:

1. An algorithm which assumes no prior knowledge and which can be used for any compression task, text or otherwise.
2. An algorithm which has some knowledge or prior probabilities and is trained for a specific compression task (such as compressing English text).
3. An algorithm which is specifically trained with respect to a corpus which corresponds to a class which we want to predict. This is very closely related to Kolmogorov Similarity Metrics.

The question arises as to whether all or any of these compression algorithms yield meaningful features and if so why. Compression analysis for machine learning has been done in a wide variety of fields. In fact much of the research has been done by those outside of machine learning who may or may not even know they are performing machine learning and who seem to have done little research into compression and complexity analysis and have no idea why their method works, only that it does.

Compression/complexity analysis has been used in many classification tasks such as image recognition, radar signal classification, EEG classification, DNA analysis, speech recognition and some text classification tasks (Chi and Kong, 1998; Zhang, Hu, and Jin, 2003; Bhattacharya, 2000; Menconi, Benci, and Buiatti, 2008; Frank, Chui, and Witten, 2000; Dalkilic et al., 2006; Seaward and Saxton, 2007; Seaward, Inkpen, and Nayak, 2008).

The method proposed here is different then those which use Kolmogorov Complexity measures to compute the distance between the object to be classified and a corpus of training data. As this is intrinsic plagiarism analysis there is no set of documents for which we can find a similarity metric. We can only compare local text to the global document. We can use a statistical compression algorithm and this is somewhat related to similarity metrics but it is not the same. We are not explicitly using the concept that "like compresses with like". Moreover, such compression measures can be used in a variety of machine learning algorithms such as support vector machines, neural networks and decision trees. We can also use boosting and meta algorithms such as bagging and ADAboost. What we are doing is finding a measure for each different distribution as to how well its complexity can be described by the compression algorithm.

6 Using Compression to Estimate Complexity

Suppose we have a statistical compression model which has been trained on a variety of English text and we compress two text samples and find that one compresses much more than the other. This means that one text was much more alike to general English text than the other was.

Now suppose we extract the noun representation of both of those texts and compress them using a statistical compression algorithm which has been trained on noun representations of English text. The one which compresses the most is closer to the normal noun distributions of English text.

What if we use a compression algorithm that has no prior training such as Lempel-Ziv? Is it still meaningful? The answer is

Classifier	Complexity features	Plagiarism	Recall	Precision	F-measure
SVM	no	yes	0.651	0.538	0.589
	no	no	0.615	0.719	0.663
	yes	yes	0.671	0.521	0.587
	yes	no	0.617	0.752	0.678
Neural network	no	yes	0.619	0.510	0.559
	no	no	0.593	0.695	0.640
	yes	yes	0.670	0.548	0.603
	yes	no	0.626	0.737	0.677

Table 1: Results on using feature sets with and without complexity features with SVM and Neural Networks.

yes because we still have an idea of the complexity of the distribution of nouns. While it does not directly relate to the norms of the English language, it is still a meaningful measure of the complexity of that distribution. It relates the noun distribution to some distribution which can be most efficiently compressed by that compression algorithm (even if we do not know the distribution).

Research has shown this holds true (Seaward and Saxton, 2007; Seaward, Inkpen, and Nayak, 2008). Dalkilic et al. (2006) have shown that Lempel-Ziv compression of text can be used to distinguish authentic text from non-authentic or computer generated text. They show that the compressibility of real texts is different than that of computer generated "nonsense" texts due to topic adherence. The idea is that when one writes a coherent text, ideas and words are repeated to increase readability.

With respect to text and compression, de Marcken theorizes that language learning is essentially a compression problem (De Marcken, 1996). If one has a great deal of knowledge about a language then one can build a model which maximizes the compressibility of text written in that language. Thus the compressibility of a text is a measure of how closely related the compression algorithm is to the text representation.

7 Experimental Results

The PAN 09 intrinsic plagiarism competition corpus consisted of 3091 annotated texts for training and 3091 texts for testing purposes (initially released unannotated). We extracted normalized counts and complexity counts for the following word classes:

Nouns	Stopwords
Verbs	Topic words
Pronouns	Common words
Adjectives	Passive words
Adverbs	Active words
Prepositions	Word length

Features were extracted locally and globally and the standard deviation amongst the local features was also computed. Zlib was used for compression.

A 50/50 training/test split was used on the training set to analyze the performance gained from adding complexity measures. The results were repeated for 10 random splits and averaged. Two classifiers were used – Support Vector Machine (SVM) and a Neural Network. Recall and precision are calculated per text chunk not per character (see Table 1).

For many classifiers tested such as regressions trees and support vector machines the F-measure performance gained by using complexity features was less than 2%. The neural network showed the most improvement with complexity measures as F-measure was increased 3.7-4.4%.

Previous classification tasks such as authorship attribution and spam filtering showed better results. The problem, as it was discovered, was the high degree of granularity required by the task. Complexity analysis does not do well with short text.

Using various feature selection tools it was found that complexity features and normalized count features were found in equal numbers in the highest ranked features. For example the top 10 features as determined by a Chi-squared feature evaluator is shown below.

As one can see in Table 2, 6 out of the 10 top ranked features are complexity features. This indicates that complexity fea-

Rank	Feature
1	Adjective complexity (l)
2	Adjective count (gsd)
3	Topic word complexity (g)
4	Verb word complexity (g)
5	Passive word complexity (g)
6	Active word complexity (g)
7	Preposition count (g)
8	Stop word count (gsd)
9	Avg. word length per sentence (gsd)
10	Topic word complexity (l)

Table 2: Top 10 ranked features for the intrinsic plagiarism task as calculated by a Chi-squared feature evaluator. l=local, g=global, gsd=global stantard deviation

tures are able to discriminate plagiarized vs. non-plagiarized passages as well as or better than normalized count features.

8 Conclusion

We introduce using compression to find features based on Kolmogorov complexity measures. We show why compression of text and word distributions results in meaningful features. Results in using complexity analysis in intrinsic plagiarism detection are promising. Performance is increased by a small amount and it seems as though complexity is not contributing to over fitting. More research needs to be done in using compression models which have prior knowledge of the language to be analyzed and/.or the prior probabilities of word classes. This would result in more meaningful complexity features which would likely aid in the difficult task of intrinsic plagiarism detection.

References

Bhattacharya, J. 2000. Complexity analysis of spontaneous EEG. *Acta Neurobiologiae Experimentalis*, 60(4):495–501.

Chi, Z. and J. Kong. 1998. Image content classification using a block Kolmogorov complexity measure. In *Proceedings of the Fourth International Conference on Signal Processing ICSP 1998*, Beijing, China.

Dalkilic, M. M., W. T. Clark, J. C. Costello, and Radivojac P. 2006. Compression to Identify Classes of Inauthentic Texts. In *Proceedings of the SIAM International Conference on Data Mining SDM 2006*, Bethesda, MD.

De Marcken, C. 1996. *Unsupervised language acquisition.* Phd thesis, Michigan Institute of Technology. http://www.demarcken.org/carl/papers/PhD.pdf.

Frank, E., C. Chui, and I. H. Witten. 2000. Text categorization using compression models. In *Proceedings of DCC-00, IEEE Data Compression Conference*, pages 200–209, Snowbird, USA. IEEE Computer Society Press.

Li, M. and P. Vitanyi. 1997. *An Introduction to Kolmogorov Complexity and its Applications.* Springer Verlag, Berlin, second edition.

Menconi, G., V. Benci, and M. Buiatti. 2008. Data compression and genomes: a two-dimensional life domain map. *Journal of Theoretical Biology*, 253(2):281–288.

Seaward, L., D. Inkpen, and A. Nayak. 2008. Using the Complexity of the Distribution of Lexical Elements as a Feature in Authorship Attribution. In *Proceedings of 6th International Conference on Language Resources and Evaluation LREC*, Marrakech, Morocco.

Seaward, L. and L. V. Saxton. 2007. Filtering spam using Kolmogorov complexity measures. In *Proceedings of the 2007 IEEE International Symposium on Data Mining and Information Retrieval (DMIR-07)*, Niagara Falls.

Stamatatos, E., N. Fakotakis, and G. Kokkinakis. 2000. Automatic Text Categorization in Terms of Genre and Author. *Computational Linguistics*, 26(4):461–485.

Stein, B. and Sven Meyer zu Eissen. 2007. Intrinsic plagiarism analysis with meta-learning. In *Proceedings of the SIGIR 2007 International Workshop on Plagiarism Analysis, Authorship Identification, and Near-Duplicate Detection, PAN 2007*, Amsterdam, Netherlands.

Zhang, G., L. Hu, and W. Jin. 2003. Complexity feature extraction of radar emitter signals. In *Environmental Electromagnetics, 2003. Proceedings of the Asia-Pacific Conference on Environmental Electromagnetics CEEM 2003*, pages 495–498.

Ordinal measures in authorship identification*

Liviu P. Dinu
University or Bucharest, Faculty of Mathematics and Computer Science, 14 Academiei, Bucharest, Romania, ldinu@funinf.cs.unibuc.ro

Marius Popescu
University or Bucharest, Faculty of Mathematics and Computer Science, 14 Academiei, Bucharest, Romania, mpopescu@phobos.cs.unibuc.ro

Abstract: The goal of this paper is to compare a set of distance/similarity measures, regarding theirs ability to reflect stylistic similarity between authors and texts. To assess the ability of these distance/similarity functions to capture stylistic similarity between texts, we tested them in one of the most frequently employed multivariate statistical analysis settings: cluster analysis. The experiments are done on a corpus of 30 English books written by British, American and Australian writers.
Keywords: authorship identification, ordinal measures

1 Introduction

The authorship identification problem is an ancient and omnipresent challenge, and almost in every culture there are a lot of disputed works (Shakespeare's plays, Moliere vs. Corneille (Labbe and Labbe, 2001), Federalist Papers (Mosteller and Wallace, 2007), etc.). The problem of authorship identification is based on the assumption that there are stylistic features that help distinguish the real author from any other possibility. Literary-linguistic research is limited by the human capacity to analyze and combine a small number of text parameters, to help solve the authorship problem. We can surpass limitation problems using computational methods, which allow us to explore various text parameters and characteristics and their combinations. Using these methods (van Halteren et al., 2005) have shown that every writer has a unique fingerprint regarding language use. The set of language use characteristics - stylistic, lexical, syntactic - form the human stylom.

Because in all computational stylistic studies/approaches, a process of comparison of two or more texts is involved, in a way or another, there was always a need for a distance/similarity function to measure similarity (or dissimilarity) of texts from the stylistic point of view. These measures vary a lot, and in the last years a series of different techniques were used in authorship identification: approaches based on string kernel (Dinu, et al., 2008), SVM based on function words frequencies (Koppel et. al., 2007), standard distances or ordinal distances (Popescu and Dinu, 2008).

The goal of this paper is to compare a set of distance/similarity measures, regarding theirs ability to reflect stylistic similarity between texts.

As style markers we have used the function words frequencies. Function words are generally considered good indicators of style because their use is very unlikely to be under the conscious control of the author and because of their psychological and cognitive role (Chung and Pennebaker, 2007). Also function words prove to be very effective in many author attribution studies.

The distance/similarity between two texts will be measured as distance/similarity between the function words frequencies corresponding to the respective texts. For this study we selected some similarity/distance measures. We started with the most natural distance/similarity measures: euclidean distance and (taking into account the statistical nature of data) Pearson's correlation coefficient. Since function words frequencies can also be viewed as ordinal variables, we also considered for comparison some specific similarity measures: Spearman's rank-order coefficient, Spearman's footrule, Goodman and Kruskal's gamma, Kendall's tau.

To assess the ability of these distance/similarity functions to capture stylistic similarity between texts, we have tested them in one of the most frequently employed multivariate statistical analysis settings: cluster

* Research supported by CNCSIS, PN2-Idei project 228

analysis. Clustering is a very good test bed for a distance/similarity measure behavior. We plugged the distance/similarity measures selected for comparison into a standard hierarchical clustering algorithm and applied it to a collection of 30 nineteenth century English books. The family trees thus obtained revealed a lot about the distance/similarity measures behavior.

The main finding of our comparison is that the similarity measures that treat function words frequencies as ordinal variables performed better than the others distance/similarity measures. Treating function words frequencies as ordinal variables means that in the calculation of distance/similarity function the ranks of function words according to their frequencies in text will be used rather than the actual values of these frequencies. Usage of the ranking of function words in the calculation of the distance/similarity measure instead of the actual values of the frequencies may seem as a loss of information, but we consider that the process of ranking makes the distance/similarity measure more robust acting as a filter, eliminating the *noise* contained in the values of the frequencies. The fact that a specific function word has the rank 2 (is the second most frequent word) in one text and has the rank 4 (is the fourth most frequent word) in another text can be more relevant than the fact that the respective word appears 34% times in the first text and only 29% times in the second.

In the next section we present the distance/similarity measures involved in the comparison study, section 3 briefly describes the cluster analysis, and in section 4 and 5 are presented the experiments, the results obtained, and suggestions for future work.

2 Similarity Measures

If we treat texts as random variables whose values are the frequencies of different words in the respective texts, then various statistical correlation measures can be used as similarity measures between that texts. For two texts X and Y and a fixed set of words $\{w_1, w_2, \ldots, w_n\}$ let denote by x_1 the relative frequency of w_1 in X, by y_1 the relative frequency of w_1 in Y and so on by x_n the relative frequency of w_n in X, by y_n the relative frequency of w_n in Y.

The *Pearson's correlation coefficient* is:

$$r = \frac{\sum_{i=1}^{n} \left(\frac{x_i - \overline{x}}{s_x}\right)\left(\frac{y_i - \overline{y}}{s_y}\right)}{n-1}$$

where $\overline{x}$ is the mean of X, $\overline{y}$ the mean of Y, s_x and s_y are the standard deviation of X, Y, respectively (Upton and Cook, 2008). The correlation coefficient measures the tendency of two variables to change in value together (i.e., to either increase or decrease). r is related with the Euclidean distance, the $\sqrt{2(1-r)}$ being the Euclidean distance between the standardized versions of X and Y.

The random variables X, Y representing texts can also be treated as ordinal data, in which data is ordered but cannot be assumed to have equal distance between values. In this case the values of X (and respectively Y) will be the ranks of words $\{w_1, w_2, \ldots, w_n\}$ according to their frequencies in text X rather than of the actual values of these frequencies. The most common correlation statistic for ordinal data is *Spearman's rank-order coefficient* (Upton and Cook 2008):

$$r_{sc} = 1 - \frac{6}{n(n^2-1)} \sum_{i=1}^{n} (x_i - y_i)^2$$

To be noted that, this time, x_i, y_i are ranks and actually, the Spearman's rank-order coefficient is the Pearson's correlation coefficient applied to ranks. The Spearman's footrule is the l_1-version of Spearman's rank-order coefficient:

$$r_{sf} = 1 - \frac{3}{n^2-1} \sum_{i=1}^{n} |x_i - y_i|$$

Another set of correlation statistics for ordinal data are based on the number of concordant and discordant pairs among two variables. The number of concordant pairs among two variables X and Y is $P = |\{(i,j) : 1 \leq i < j \leq n, (x_i - x_j)(y_i - y_j) > 0\}|$. Similarly, the number of discordant pairs is $Q = |\{(i,j) : 1 \leq i < j \leq n, (x_i - x_j)(y_i - y_j) < 0\}|$.

Goodman and Kruskal's gamma(Upton and Cook 2008) is defined as:

$$\gamma = \frac{P - Q}{P + Q}$$

Kendall developed several slightly different types of ordinal correlation as alternatives to gamma. *Kendall's tau-a*(Upton and

Cook 2008) is based on the number of concordant versus discordant pairs, divided by a measure based on the total number of pairs (n = the sample size):

$$\tau_a = \frac{P - Q}{\frac{n(n-1)}{2}}$$

Kendall's tau-b(Upton and Cook 2008) is a similar measure of association based on concordant and discordant pairs, adjusted for the number of ties in ranks.It is calculated as $(P - Q)$ divided by the geometric mean of the number of pairs not tied on X (X_0) and the number of pairs not tied on Y (Y_0):

$$\tau_b = \frac{P - Q}{\sqrt{(P + Q + X_0)(P + Q + Y_0)}}$$

All the above three correlation statistics are very related, if n is fixed and X and Y have no tied, then P, X_0 and Y_0 are completely determined by n and Q.

3 Clustering Analysis

An agglomerative hierarchical clustering algorithm (Duda et. al. 2001) arranges a set of objects in a family tree (dendogram) according to their similarity, similarity which in its turn is given by a distance function defined on the set of objects. The algorithm initially assigns each object to its own cluster and then repeatedly merges pairs of clusters until the whole tree is formed. At each step the pair of nearest clusters is selected for merging. Various agglomerative hierarchical clustering algorithms differ in the way in which they measure the distance between clusters. Note that although a distance function between objects exists, the distance measure between clusters (set of objects) remains to be defined. In our experiments we used the *complete linkage* distance between clusters, the maximum of the distances between all pairs of objects drawn from the two clusters (one object from the first cluster, the other from the second).

4 Experiments

In Popescu and Dinu (2009) we have compared the set of distance/similarity measures described here on a collection of 21 nineteenth century English books written by 10 different authors and spanning a variety of genre (the same set of books were used by Koppel et al. (2007) in their authorship verification experiments). The experiments have shown that the similarity measures that treat function words frequencies as ordinal variables (Spearman's rank-order coefficient, Spearman's footrule, Goodman and Kruskal's gamma, Kendall's tau) performed better than the distance/similarity measures that use the actual values of function words frequencies (Euclidean distance, Pearson's correlation coefficient).

Group	Author	Book
American Novelists	Hawthorne	Dr. Grimshawe's Secret
		House of Seven Gables
	Melville	Redburn
		Moby Dick
	Cooper	The Last of the Mohicans
		The Spy
		Water Witch
American Essayists	Thoreau	Walden
		A Week on Concord
	Emerson	Conduct Of Life
		English Traits
British Playwrights	Shaw	Pygmalion
		Misalliance
		Getting Married
	Wilde	An Ideal Husband
		Woman of No Importance
Bronte Sisters	Anne	Agnes Grey
		Tenant Of Wildfell Hall
	Charlotte	The Professor
		Jane Eyre
	Emily	Wuthering Heights
Australian Novelists	B. Baynton	Bush Studies
		Human Toll
	Henry Lawson	Joe Wilson and His Mates
		On the Track
		While the Billy Boils
	Miles Franklin	My Brilliant Career
		Some Everyday Folk and Dawn
		Up the Country: A Saga of...
		Back to Bool Bool

Table 1: The books used in experiments

The aim of the actual experiments was two-folded. Firstly we wanted to see if the findings in Popescu and Dinu (2009) are confirmed in the case of a larger set (more authors, more books) and secondly to further investigate the ability of some of the similarity measures (Spearman's rank-order coefficient, Goodman and Kruskal's gamma, Kendall's tau) to distinguish between the different nationality of English language writers by adding to the data set works of Australian writers from the same period. To the original data set of Koppel et al. (2007) we added 9 works of three Australian authors from the same period, resulting a data set of 30 books and 13 authors (Table 1).

To perform the experiments, a set of words must be fixed. The most frequent function words may be selected or other criteria may be used for selection. In all our experiments we used the set of function words identified by Mosteller and Wallace (2007) as good candidates for author-attribution stud-

ies. We used the agglomerative hierarchical clustering algorithm coupled with the various distance similarity function employed in the comparison to cluster the works in Table 1.

The dendrograms obtained sustain the results of Popescu and Dinu (2009). The resulted dendrograms for Euclidean distance and Pearson's correlation coefficient (not shown because of lack of space) are very similar, which is no surprise taking into account the close relation between the two measures (see section 2.1). The problem of these family trees is that the works of Melville are not grouped together: one being clustered with the essays of Thoreau (Moby Dick) and the other with the novels of Hawthorne. Also, "My Brilliant Career" of M. Franklin is clustered with the novels of Charlotte Bronte. Apart from authorship relation, the dendrograms reflect no other stylistic relation between the works (like grouping the works according to genre or nationality of the authors: American / English / Australian).

Spearman's rank-order coefficient, Goodman and Kruskal's gamma and Kendall's tau produced the same dendrogram (modulo the scale).Figure 1 shows the dendrogram for Kendall's tau. The dendrogram is perfect: all works are clustered according to theirs author. The nationality of the authors is not reflected in the dendrogram (the authors with the same nationality are not clustered together).

We performed a series of experiments to test in which cases the nationality of the authors can be revealed by a stylistic similarity measure. If only British and Australian writers are selected, the Kendall's tau produced the dendrogram presented in Figure 2. As can be seen the first two branches correspond to the nationality of the authors: British writers on upper branch, Australian writers on lower branch. The same thing happen when British and American writers are selected. Again, the writers are clustered according to their nationality: this time, the British writers on lower branch and American writers on upper branch. But when the subset of American and Australian writers is clustered using Kendall's tau, the nationality of the writers is no longer reflected in the family tree produced. The works of each author are clustered together, but there are no clear branches corresponding to the two nationalities.

5 Future Work

In this paper we have compared a set of measures, regarding theirs ability to reflect stylistic similarity between texts. In future work it would be interesting to compare these measures to other possible similarity measures. If the frequencies of different words in the texts are treated as probability distributions instead as random variables, specific measures can be applied: Kullback-Liebler Divergence or Cross Entropy.

References

C. K. Chung, and J. W. Pennebaker. 2007. The psychological function of function words. In K. Fiedler, ed., *Social communication: Frontiers of social psychology*, 343−359. Psychology Press, New York.

L.P. Dinu, M. Popescu and A. Dinu. 2008. Authorship Identification of Romanian Texts with Controversial Paternity. *Proc. LREC 2008*, Marrakech, Morocco.

R. O. Duda, P. E. Hart, and D. G. Stork. 2001. *Pattern Classification* (2nd ed.). Wiley-Interscience Publication.

H. van Halteren, M. Haverkort, H. Baayen, A. Neijt, and F. Tweedie. 2005. New machine learning methods demonstrate the existence of a human stylome. *Journal of Quantitative Linguistics*, 12:65−77.

M. Koppel, J. Schler, and E. Bonchek-Dokow. 2007. Measuring differentiability: Unmasking pseudonymous authors. *J. of Machine Learning Research*, 8,1261 −1276.

C. Labbe and D. Labbe. 2006. A tool for literary studies: Intertextual distance and tree classification. *Literary and Linguistic Computing*, 21(3):311−326.

F. Mosteller and D.L. Wallace. 2007. *Inference and Disputed Authorship: The Federalist*. CSLI Publications, Stanford.

M. Popescu, L.P.Dinu, 2008. Rank Distance as a Stylistic Similarity. *Proceedings COLING 2008*, Manchester, UK

M. Popescu, L.P.Dinu, 2009. Comparing Statistical Similarity Measures for Stylistic Multivariate Analysis. *Proceedings RANLP 2009*, Borovets, Bulgaria

G. Upton and I. Cook. 2008. *A Dictionary of Statistics*. Oxford Univ. Press, Oxford.

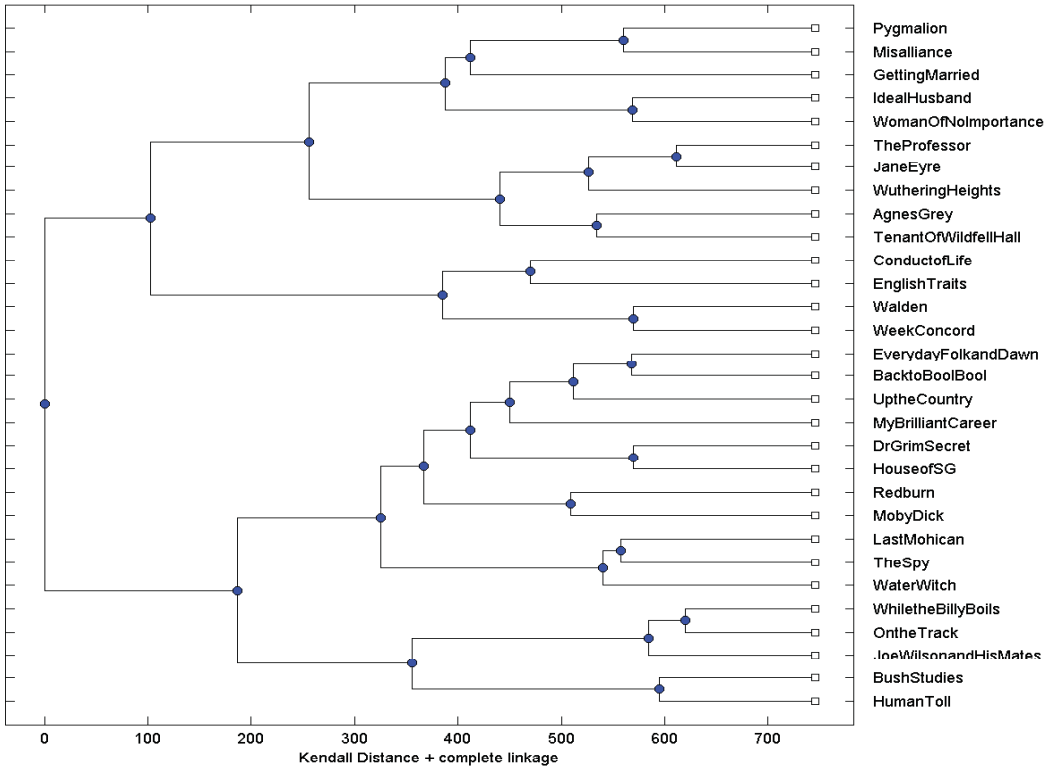

Figure 1: Dendrogram of 30 nineteenth century English books (Kendal's tau)

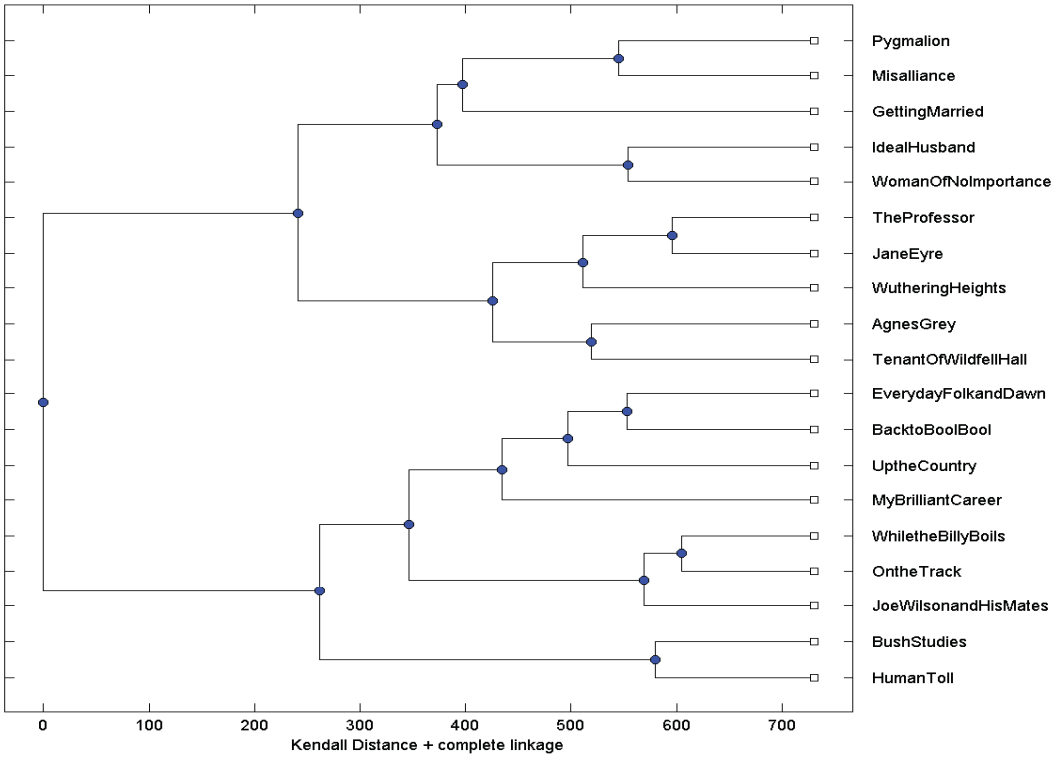

Figure 2: Dendrogram of British and Australian writers (Kendal's tau)

“Counter plagiarism detection software” and “Counter counter plagiarism detection” methods.

Yurii Palkovskii
SkyLine, Inc.
Ukraine, Mayakovskogo 15, 66 p.code: 10014
palkovskiy@yandex.ru

Abstract: This article describes the basic principles of counter plagiarism detection software, its methods and the countermeasures that can be applied in order to effectively oppose such software.
Keywords: plagiarism detection, plagiarism, counter plagiarism detection software, plagiarism detection methods.

1 Plagiarism detection as a complex problem in Higher Educational Institutions

1.1 The necessity of plagiarism detection complex solutions in Ukrainian education

Plagiarism detection and prevention becomes an important issue in Ukrainian higher educational institutions. There exists a number of reasons that deepen the academic dishonesty problem. The rapid spread of information technology and the Internet in particular has reached the maximum degree only during the recent years of 2004-2008. Multiplied by the insufficient experience of the teaching personel in relation to the Internet technologies and plagiarism detection methods in particular, it resulted in lack of control over the students’ written assignments quality. This latest tend caused the development and intruduction of different plagiarism detection and prevention software solutions and their wide integration into the study process in different educational establishmments all over the country.

2 “Counter plagiarism detection software” and “Counter counter plagiarism detection” methods

2.1 “Counter plagiarism detection software”

One of the most interesting aspects of the plagiarism detection on the territory of the former USSR is the fact that general students’ IT competence is much higher than that of the teachers. This caused the number of cases of special software being developed by students as a countermeasure to fight plagiarism detection services and software solutions. One of the examples to mention is: “AntiPlagiatKiller”. The exact title of the software in Russian is “антиплагиат киллер”. It is a special text post-processing application that uses effective algorithms that “shuffle” or otherwise scramble the target text using different methods to render the fingerprint search useless by “breaking up” the signatures that are generated for indexing and search.

2.2 Counter plagiarism detection algorithms

Some principles used in: “AntiPlagiatKiller” are similar to the ones used in the “Random Plagiarist” that was developed for the PAN09 plagiarism detection competition to provide obfuscated plagiarism sections, some of these algorithms are generic and some are relevant to Slavonic languages only. for example – the so called “Cyrillic to English substitution”. The particular method implies a substitution of some or all of the particular character or characters from the native text alphabet for their Latin equivalent that has the same visual shape. This method breaks the fingerprint signature and remains visually undetectable if the investigation is done by a human reviewer. For example:

Russian “o” – Latin (English) “o”
Ukrainian “x” – Latin (English) “x”

Example: **T**o**ast**

Stein, Rosso, Stamatatos, Koppel, Agirre (Eds.): PAN'09, pp. 67-68, 2009.

The letter “o” in the word “Toast” is not English, but Russian.

This particular method can be implemented and is massively exploited in a particular software environment, for example Microsoft Word. It cannot be implemented in plain text and different document formats that does not support the required functionality.

This applies to a number of Languages that share the same graphical alphabet (or partially share): Russian, Ukrainian, Belarusian and some others. This method can be potentially applied to any relevant languages that share at least a pair of visually identical characters.

The other generic method that is widely used by the modification of “anti-plagiarism-killer” is even more original, it is called – “white link-character insertion”. The idea of this obfuscation method is simple and effective against the automated plagiarism detection systems and again visually hard to spot – a space between any words is filled with a single character or a pair of random characters that are later colored in white. This completely distorts the semantic structure of the sentence but remains visually undetectable. To illustrate this method the following example can be shown:

Original text: **I like apples**
Scrambled text: **I**x**like**x**apples**

The scrambled text contains character “x” (and for the sake of the example it is colored in grey instead of white) as a “white link-character” that is inserted between the words. It has white as its background color that makes it absolutely invisible to visually detect.

The appearance of such software and its wide usage among students forces the developers of the plagiarism detection solutions to present a number of additional algorithms to detect such “shuffling” done to the text and add a special alert to the final originality report to inform the reviewer about the possible usage of the “counter plagiarism detection” algorithms.

The existence of such software points to the necessity of the development more sophisticated plagiarism detection systems. The first generation of such solutions uses the fingerprinting method to detect the cases of potential plagiarism on the basis of the fingerprint taken from the sentence – that is it uses a sentence to count a hash to be later used in the search. The “Counter plagiarism detection software” is pretty effective in distorting the text to the degree when fingerprints cannot be effectively used any longer and thus the need to use another algorithm is brought to life.

2.3 Counter counter plagiarism detection algorithms

The implementation of each algorithm that will detect the cases of deliberate obfuscation is rather straightforward in case the obfuscation mechanism deals with the text meta characteristics – color, character set or encoding. Every implementation of such algorithm must be targeted against the specific obfuscation method with two possible output results – either neutralization of the counter plagiarism detection algorithm to remove the factors that prevent normal analysis or reporting the number and scope of the particular occurrences (or suspected occurrences) in the analyzed text.

References:

1. Antiplagiatikiller download page: http://www.datalive.ru/2007/08/20/anti_plagiat_killer_borba_s_antiplagiatom_postudencheski.html

2. Official reaction to Antiplagiatikiller: http://www.antiplagiat.ru/forum/default.aspx?g=posts&t=223

3. Internet board that contains the methods on how to check plagiarism detection systems: http://www.antiplagiat.ru/forum/default.aspx?g=topics&f=5

Author Index