Proceedings of the 23rd International Conference on Computational Linguistics, COLING 2010
Beijing, China, ©ACL 2010.

An Evaluation Framework for Plagiarism Detection

Martin Potthast **Benno Stein**
Web Technology & Information Systems
Bauhaus-Universität Weimar
{martin.potthast, benno.stein}@uni-weimar.de

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

Abstract

We present an evaluation framework for plagiarism detection.[1] The framework provides performance measures that address the specifics of plagiarism detection, and the PAN-PC-10 corpus, which contains 64 558 *artificial* and 4 000 *simulated* plagiarism cases, the latter generated via Amazon's Mechanical Turk. We discuss the construction principles behind the measures and the corpus, and we compare the quality of our corpus to existing corpora. Our analysis gives empirical evidence that the construction of tailored training corpora for plagiarism detection can be automated, and hence be done on a large scale.

1 Introduction

The lack of an evaluation framework is a serious problem for every empirical research field. In the case of plagiarism detection this shortcoming has recently been addressed for the first time in the context of our benchmarking workshop PAN [15, 16]. This paper presents the evaluation framework developed in the course of the workshop. But before going into details, we survey the state of the art in evaluating plagiarism detection, which has not been studied systematically until now.

1.1 A Survey of Evaluation Methods

We have queried academic databases and search engines to get an overview of all kinds of contributions to automatic plagiarism detection. Altogether 275 papers were retrieved, from which 139 deal with plagiarism detection in text, 123 deal with plagiarism detection in code, and 13 deal with other media types. From the papers related to text and code we analyzed the 205 which present evaluations. Our analysis covers the following aspects: experiment tasks, performance measures, underlying corpora, and, whether comparisons to other plagiarism detection approaches were conducted. Table 1 summarizes our findings.

Table 1: Summary of the plagiarism detection evaluations in 205 papers, from which 104 deal with text and 101 deal with code.

Evaluation Aspect	Text	Code
Experiment Task		
local collection	80%	95%
Web retrieval	15%	0%
other	5%	5%
Performance Measure		
precision, recall	43%	18%
manual, similarity	35%	69%
runtime only	15%	1%
other	7%	12%
Comparison		
none	46%	51%
parameter settings	19%	9%
other algorithms	35%	40%

Evaluation Aspect	Text	Code
Corpus Acquisition		
existing corpus	20%	18%
homemade corpus	80%	82%
Corpus Size [# documents]		
$[1, 10)$	11%	10%
$[10, 10^2)$	19%	30%
$[10^2, 10^3)$	38%	33%
$[10^3, 10^4)$	8%	11%
$[10^4, 10^5)$	16%	4%
$[10^5, 10^6)$	8%	0%

With respect to the experiment tasks the majority of the approaches perform overlap detection by exhaustive comparison against some locally stored document collection—albeit a Web retrieval scenario is more realistic. We explain this shortcoming by the facts that the Web cannot be utilized easily as a corpus, and, that in the case of code plagiarism the focus is on collusion detection in student courseworks. With respect to performance measures the picture is less clear: a manual result evaluation based on similarity measures is used about the same number of times for text (35%), and even more often for code (69%), as an automatic computation of precision and recall. 21% and 13% of the evaluations on text and code use custom measures or examine only the de-

[1] The framework is available free of charge at `http://www.webis.de/research/corpora`.

tection runtime. This indicates that precision and recall may not be well-defined in the context of plagiarism detection. Moreover, comparisons to existing research are conducted in less than half of the papers, a fact that underlines the lack of an evaluation framework.

The right-hand side of Table 1 overviews two corpus-related aspects: the use of existing corpora versus the use of handmade corpora, and the size distribution of the used corpora. In particular, we found that researchers follow two strategies to compile a corpus. Small corpora (<1 000 documents) are built from student courseworks or from arbitrary documents into which plagiarism-alike overlap is manually inserted. Large corpora (>1 000 documents) are collected from sources where overlap occurs more frequently, such as rewritten versions of news wire articles, or from consecutive versions of open source software. Altogether, we see a need for an open, commonly used plagiarism detection corpus.

1.2 Related Work

There are a few surveys about automatic plagiarism detection in text [7, 8, 14] and in code [12, 17, 19, 20]. These papers, as well as nearly all papers of our survey, omit a discussion of evaluation methodologies; the following 4 papers are an exception.

In [21] the authors introduce graph-based performance measures for code plagiarism detection that are intended for unsupervised evaluations. We argue that evaluations in this field should be done in a supervised manner. An aside: the proposed measures have not been adopted since their first publication. In [15] we introduce preliminary parts of our framework. However, the focus of that paper is less on methodology but on the comparison of the detection approaches that were submitted to the first PAN benchmarking workshop. In [9, 10] the authors report on an unnamed corpus that comprises 57 cases of simulated plagiarism. We refer to this corpus as the Clough09 corpus; a comparison to our approach is given later on. Finally, a kind of related corpus is the METER corpus, which has been the only alternative for the text domain up to now [11]. It comprises 445 cases of text reuse among 1 716 news articles. Although the corpus can be used to evaluate plagiarism detection its design does not support this task. This is maybe the reason why it has not been used more often. Furthermore, it is an open question whether or not cases of news reuse differ from plagiarism cases where the plagiarists strive to remain undetected.

1.3 Contributions

Besides the above survey, the contributions of our paper are threefold: Section 2 presents formal foundations for the evaluation of plagiarism detection and introduces three performance measures. Section 3 introduces methods to create artificial and simulated plagiarism cases on a large scale, and the PAN-PC-10 corpus in which these methods have been operationalized. Section 4 then compares our corpus with the Clough09 corpus and the METER corpus. The comparison reveals important insights for the different kinds of text reuse in these corpora.

2 Plagiarism Detection Performance

This section introduces measures to quantify the precision and recall performance of a plagiarism detection algorithm; we present a micro-averaged and a macro-averaged variant. Moreover, the so-called detection granularity is introduced, which quantifies whether the contiguity between plagiarized text passages is properly recognized. This concept is important: a low granularity simplifies both the human inspection of algorithmically detected passages as well as an algorithmic style analysis within a potential post-process. The three measures can be applied in isolation but also be combined into a single, overall performance score. A reference implementation of the performance measures is distributed with our corpus.

2.1 Precision, Recall, and Granularity

Let d_{plg} denote a document that contains plagiarism. A *plagiarism case* in d_{plg} is a 4-tuple $s = \langle s_{\text{plg}}, d_{\text{plg}}, s_{\text{src}}, d_{\text{src}} \rangle$, where s_{plg} is a plagiarized passage in d_{plg}, and s_{src} is its original counterpart in some source document d_{src}. Likewise, a *plagiarism detection* for document d_{plg} is denoted as $r = \langle r_{\text{plg}}, d_{\text{plg}}, r_{\text{src}}, d'_{\text{src}} \rangle$; r associates an allegedly plagiarized passage r_{plg} in d_{plg} with

a passage r_{src} in d'_{src}. We say that r *detects* s iff $r_{\text{plg}} \cap s_{\text{plg}} \neq \emptyset$, $r_{\text{src}} \cap s_{\text{src}} \neq \emptyset$, and $d'_{\text{src}} = d_{\text{src}}$. With regard to a plagiarized document d_{plg} it is assumed that different plagiarized passages of d_{plg} do not intersect; with regard to detections for d_{plg} no such restriction applies. Finally, S and R denote sets of plagiarism cases and detections.

While the above 4-tuples resemble an intuitive view of plagiarism detection we resort to an equivalent, more concise view to simplify the subsequent notations: a document d is represented as a set of references to its characters $\mathbf{d} = \{(1, d), \ldots, (|d|, d)\}$, where (i, d) refers to the i-th character in d. A plagiarism case s can then be represented as $\mathbf{s} = \mathbf{s}_{\text{plg}} \cup \mathbf{s}_{\text{src}}$, where $\mathbf{s}_{\text{plg}} \subseteq \mathbf{d}_{\text{plg}}$ and $\mathbf{s}_{\text{src}} \subseteq \mathbf{d}_{\text{src}}$. The characters referred to in $\mathbf{s}_{\text{plg}}$ and $\mathbf{s}_{\text{src}}$ form the passages s_{plg} and s_{src}. Likewise, a detection r can be represented as $\mathbf{r} = \mathbf{r}_{\text{plg}} \cup \mathbf{r}_{\text{src}}$. It follows that r detects s iff $\mathbf{r}_{\text{plg}} \cap \mathbf{s}_{\text{plg}} \neq \emptyset$ and $\mathbf{r}_{\text{src}} \cap \mathbf{s}_{\text{src}} \neq \emptyset$. Based on these representations, the micro-averaged precision and recall of R under S are defined as follows:

$$prec_{\text{micro}}(S, R) = \frac{|\bigcup_{(s,r)\in(S\times R)}(\mathbf{s} \sqcap \mathbf{r})|}{|\bigcup_{r\in R} \mathbf{r}|}, \quad (1)$$

$$rec_{\text{micro}}(S, R) = \frac{|\bigcup_{(s,r)\in(S\times R)}(\mathbf{s} \sqcap \mathbf{r})|}{|\bigcup_{s\in S} \mathbf{s}|}, \quad (2)$$

$$\text{where} \quad \mathbf{s} \sqcap \mathbf{r} = \begin{cases} \mathbf{s} \cap \mathbf{r} & \text{if } r \text{ detects } s, \\ \emptyset & \text{otherwise.} \end{cases}$$

The macro-averaged precision and recall are unaffected by the length of a plagiarism case; they are defined as follows:

$$prec_{\text{macro}}(S, R) = \frac{1}{|R|} \sum_{r\in R} \frac{|\bigcup_{s\in S}(\mathbf{s} \sqcap \mathbf{r})|}{|\mathbf{r}|}, \quad (3)$$

$$rec_{\text{macro}}(S, R) = \frac{1}{|S|} \sum_{s\in S} \frac{|\bigcup_{r\in R}(\mathbf{s} \sqcap \mathbf{r})|}{|\mathbf{s}|}, \quad (4)$$

Besides precision and recall there is another concept that characterizes the power of a detection algorithm, namely, whether a plagiarism case $s \in S$ is detected as a whole or in several pieces. The latter can be observed in today's commercial plagiarism detectors, and the user is left to combine these pieces to a consistent approximation of s. Ideally, an algorithm should report detections R in a one-to-one manner to the true cases S. To capture this characteristic we define the detection granularity of R under S:

$$gran(S, R) = \frac{1}{|S_R|} \sum_{s\in S_R} |R_s|, \quad (5)$$

where $S_R \subseteq S$ are cases detected by detections in R, and $R_s \subseteq R$ are the detections of a given s:

$$S_R = \{s \mid s \in S \wedge \exists r \in R : r \text{ detects } s\},$$
$$R_s = \{r \mid r \in R \wedge r \text{ detects } s\}.$$

The domain of $gran(S, R)$ is $[1, |R|]$, with 1 indicating the desired one-to-one correspondence and $|R|$ indicating the worst case, where a single $s \in S$ is detected over and over again.

Precision, recall, and granularity allow for a partial ordering among plagiarism detection algorithms. To obtain an absolute order they must be combined to an overall score:

$$plagdet(S, R) = \frac{F_\alpha}{\log_2(1 + gran(S, R))}, \quad (6)$$

where F_α denotes the F_α-Measure, i.e., the weighted harmonic mean of precision and recall. We suggest using $\alpha = 1$ (precision and recall equally weighted) since there is currently no indication that either of the two is more important. We take the logarithm of the granularity to decrease its impact on the overall score.

2.2 Discussion

Plagiarism detection is both a retrieval task and an extraction task. In light of this fact not only retrieval performance but also extraction accuracy becomes important, the latter of which being neglected in the literature. Our measures incorporate both. Another design objective of our measures is the minimization of restrictions imposed on plagiarism detectors. The overlap restriction for plagiarism cases within a document assumes that a certain plagiarized passage is unlikely to have more than one source. Imprecision or lack of evidence, however, may cause humans or algorithms to report overlapping detections, e.g., when being unsure about the true source of a plagiarized passage. The measures (1)-(4) provide for a sensible treatment of this fact since the set-based

passage representations eliminate duplicate detections of characters. The macro-averaged variants allot equal weight to each plagiarism case, regardless of its length. Conversely, the micro-averaged variants favor the detection of long plagiarism passages, which are generally easier to be detected. Which of both is to be preferred, however, is still an open question.

3 Plagiarism Corpus Construction

This section organizes and analyzes the practices that are employed—most of the time implicitly—for the construction of plagiarism corpora. We introduce three levels of *plagiarism authenticity*, namely, real plagiarism, simulated plagiarism, and artificial plagiarism. It turns out that simulated plagiarism and artificial plagiarism are the only viable alternatives for corpus construction. We propose a new approach to scale up the generation of simulated plagiarism based on crowdsourcing, and heuristics to generate artificial plagiarism. Moreover, based on these methods, we compile the PAN plagiarism corpus 2010 (PAN-PC-10) which is the first corpus of its kind that contains both a large number and a high diversity of artificial and simulated plagiarism cases.

3.1 Real, Simulated, and Artificial Plagiarism

Syntactically, a plagiarism case is the result of copying a passage s_{src} from a source document into another document d_{plg}. Since verbatim copies can be detected easily, plagiarists often rewrite s_{src} to obfuscate their illegitimate act. This behavior must be modeled when constructing a training corpus for plagiarism detection, which can be done at three levels of authenticity. Ideally, one would secretly observe a large number of plagiarists and use their *real plagiarism* cases; at least, one could resort to plagiarism cases which have been detected in the past. The following aspects object against this approach:

- The distribution of detected real plagiarism is skewed towards ease of detectability.
- The acquisition of real plagiarism is expensive since it is often concealed.
- Publishing real cases requires the consents from the plagiarist and the original author.
- A public corpus with real cases is questionable from an ethical and legal viewpoint.
- The anonymization of real plagiarism is difficult due to Web search engines and authorship attribution technology.

It is hence more practical to let people create plagiarism cases by "purposeful" modifications, or to tap resources that contain similar kinds of text reuse. We subsume these strategies under the term *simulated plagiarism*. The first strategy has often been applied in the past, though on a small scale and without a public release of the corpora; the second strategy comes in the form of the METER corpus [11]. Note that, from a psychological viewpoint, people who simulate plagiarism act under a different mental attitude than plagiarists. From a linguistic viewpoint, however, it is unclear whether real plagiarism differs from simulated plagiarism.

A third possibility is to generate plagiarism algorithmically [6, 15, 18], which we call *artificial plagiarism*. Generating artificial plagiarism cases is a non-trivial task if one requires semantic equivalence between a source passage s_{src} and the passage s_{plg} that is obtained by an automatic obfuscation of s_{src}. Such semantics-preserving algorithms are still in their infancy; however, the similarity computation between texts is usually done on the basis of document models like the bag of words model and not on the basis of the original text, which makes obfuscation amenable to simpler approaches.

3.2 Creating Simulated Plagiarism

Our approach to scale up the creation of simulated plagiarism is based on Amazon's Mechanical Turk, AMT, a commercial crowdsourcing service [3]. This service has gathered considerable interest, among others to recreate TREC assessments [1], but also to write and translate texts [2].

We offered the following task on the Mechanical Turk platform: *Rewrite the original text found below* [on the task Web page] *so that the rewritten version has the same meaning as the original, but with a different wording and phrasing. Imagine a scholar copying a friend's homework just before class, or imagine a plagiarist willing to use the*

Table 2: Summary of 4 000 Mechanical Turk tasks completed by 907 workers.

Worker Demographics			
Age		*Education*	
18, 19	10%	HS	11%
20–29	37%	College	30%
30–39	16%	BSc.	17%
40–49	7%	MSc.	11%
50–59	4%	Dr.	2%
60–69	1%		
n/a	25%	n/a	29%
Native Speaker		*Gender*	
yes	62%	male	37%
no	14%	female	39%
n/a	23%	n/a	24%
Prof. Writer		*Plagiarized*	
yes	10%	yes	16%
no	66%	no	60%
n/a	24%	n/a	25%

Task Statistics	
Tasks per Worker	
average	15
std. deviation	20
minimum	1
maximum	103
Work Time (minutes)	
average	14
std. deviation	21
minimum	1
maximum	180
Compensation	
pay per task	0.5 US$
rejected results	25%

original text without proper citation.

Workers were required to be fluent in English reading and writing, and they were informed that every result was to be reviewed. A questionnaire displayed alongside the task description asked about the worker's age, education, gender, and native speaking ability. Further we asked whether the worker is a professional writer, and whether he or she has ever plagiarized. Completing the questionnaire was optional in order to minimize false answers, but still, these numbers have to be taken with a grain of salt: the Mechanical Turk is not the best environment for such surveys. Table 2 overviews the worker demographics and task statistics. The average worker appears to be a well-educated male or female in the twenties, whose mother tongue is English. 16% of the workers claim to have plagiarized at least once, and if at least the order of magnitude of the latter number can be taken seriously this shows that plagiarism is a prevalent problem.

A number of pilot experiments were conducted to determine the pay per task, depending on the text length and the task completion time: for 50 US-cents about 500 words get rewritten in about half an hour. We observed that decreasing or increasing the pay per task has proportional effect on the task completion time, but not on the result quality. This observation is in concordance with earlier research [13]. Table 3 contrasts a source passage s_{src} and its rewritten, plagiarized passage s_{plg} obtained via the Mechanical Turk.

3.3 Creating Artificial Plagiarism

To create artificial plagiarism, we propose three obfuscation strategies. Given a source passage s_{src} a plagiarized passage s_{plg} can be created as follows (see Table 4):

- *Random Text Operations.* s_{plg} is created from s_{src} by shuffling, removing, inserting, or replacing words or short phrases at random. Insertions and replacements are taken from the document d_{plg} where s_{plg} is to be inserted.
- *Semantic Word Variation.* s_{plg} is created from s_{src} by replacing words by one of their synonyms, antonyms, hyponyms, or hypernyms, chosen at random. A word is kept if none of them is available.

Table 3: Example of a simulated plagiarism case s, generated with Mechanical Turk.

Source Passage s_{src}	**Plagiarized Passage** s_{plg}
The emigrants who sailed with Gilbert were better fitted for a crusade than a colony, and, disappointed at not at once finding mines of gold and silver, many deserted; and soon there were not enough sailors to man all the four ships. Accordingly, the Swallow was sent back to England with the sick; and with the remainder of the fleet, well supplied at St. John's with fish and other necessaries, Gilbert (August 20) sailed south as far as forty-four degrees north latitude. Off Sable Island a storm assailed them, and the largest of the vessels, called the Delight, carrying most of the provisions, was driven on a rock and went to pieces. [Excerpt from "*Abraham Lincoln: A History*" by John Nicolay and John Hay.]	The people who left their countries and sailed with Gilbert were more suited for fighting the crusades than for leading a settled life in the colonies. They were bitterly disappointed as it was not the America that they had expected. Since they did not immediately find gold and silver mines, many deserted. At one stage, there were not even enough man to help sail the four ships. So the Swallow was sent back to England carrying the sick. The other fleet was supplied with fish and the other necessities from St. John. On August 20, Gilbert had sailed as far as forty-four degrees to the north latitude. His ship known as the Delight, which bore all the required supplies, was attacked by a violent storm near Sable Island. The storm had driven it into a rock shattering it into pieces.

Table 4: Examples of the obfuscation strategies.

Obfuscation Examples
Original Text The quick brown fox jumps over the lazy dog.
Manual Obfuscation (by a human) Over the dog which is lazy jumps quickly the fox which is brown. Dogs are lazy which is why brown foxes quickly jump over them. A fast auburn vulpine hops over an idle canine.
Random Text Operations over The. the quick lazy dog <*context word*> jumps brown fox over jumps quick brown fox The lazy. the brown jumps the. quick dog The lazy fox over
Semantic Word Variation The quick brown dodger leaps over the lazy canine. The quick brown canine jumps over the lazy canine. The quick brown vixen leaps over the lazy puppy.
POS-preserving Word Shuffling The brown lazy fox jumps over the quick dog. The lazy quick dog jumps over the brown fox. The brown lazy dog jumps over the quick fox.

- *POS-preserving Word Shuffling.* The sequence of parts of speech in s_{src} is determined and s_{plg} is created by shuffling words at random while retaining the original POS sequence.

To generate different degrees of obfuscation the strategies can be adjusted by varying the number of operations made on s_{src}, and by limiting the range of affected phrases within s_{src}. For our corpus, the strategies were combined and adjusted to match an intuitive understanding of a "low" and a "high" obfuscation. Of course other obfuscation strategies are conceivable, e.g., based on automatic paraphrasing methods [4], but for performance reasons simple strategies are preferred at the expense of readability of the obfuscated text.

3.4 Overview of the PAN-PC-10

To compile the PAN plagiarism corpus 2010, several other parameters besides the above plagiarism obfuscation methods have been varied. Table 5 gives an overview.

The documents used in the corpus are derived from books from the Project Gutenberg.[2] Every document in the corpus serves one of two purposes: it is either used as a source for plagiarism or as a document suspicious of plagiarism. The latter documents divide into documents that actually contain plagiarism and documents that don't.

[2] http://www.gutenberg.org

Table 5: Corpus statistics of the PAN-PC-10 for its 27 073 documents and 68 558 plagiarism cases.

Document Statistics	
Document Purpose	
source documents	50%
suspicious documents	
– with plagiarism	25%
– w/o plagiarism	25%
Intended Algorithms	
external detection	70%
intrinsic detection	30%
Plagiarism per Document	
hardly (5%-20%)	45%
medium (20%-50%)	15%
much (50%-80%)	25%
entirely (>80%)	15%
Document Length	
short (1-10 pp.)	50%
medium (10-100 pp.)	35%
long (100-1000 pp.)	15%

Plagiarism Case Statistics	
Topic Match	
intra-topic cases	50%
inter-topic cases	50%
Obfuscation	
none	40%
artificial	
– low obfuscation	20%
– high obfuscation	20%
simulated (AMT)	6%
translated ({de,es} to en)	14%
Case Length	
short (50-150 words)	34%
medium (300-500 words)	33%
long (3000-5000 words)	33%

The documents without plagiarism allow to determine whether or not a detector can distinguish plagiarism cases from overlaps that occur naturally between random documents.

The corpus is split into two parts, corresponding to the two paradigms of plagiarism detection, namely external plagiarism detection and intrinsic plagiarism detection. Note that in the case of intrinsic plagiarism detection the source documents used to generate the plagiarism cases are omitted: intrinsic detection algorithms are expected to detect plagiarism in a suspicious document by analyzing the document in isolation. Moreover, the intrinsic plagiarism cases are not obfuscated in order to preserve the writing style of the original author; the 40% of unobfuscated plagiarism cases in the corpus include the 30% of the cases belonging to the intrinsic part.

The fraction of plagiarism per document, the lengths of the documents and plagiarism cases, and the degree of obfuscation per case determine the difficulty of the cases: the corpus contains short documents with a short, unobfuscated plagiarism case, resulting in a 5% fraction of plagiarism, but it also contains large documents with several obfuscated plagiarism cases of varying lengths, drawn from different source documents and resulting in fractions of plagiarism up to 100%. Since the true distributions of these parameters in real plagiarism are unknown, sensible

estimations were made for the corpus. E.g., there are more simple plagiarism cases than complex ones, where "simple" refers to short cases, hardly plagiarism per document, and less obfuscation.

Finally, plagiarism cases were generated between topically related documents and between unrelated documents. To this end, the source documents and the suspicious documents were clustered into $k = 30$ clusters using bisecting k-means [22]. Then an equal share of plagiarism cases were generated for pairs of source documents and suspicious documents within as well as between clusters. Presuming the clusters correspond to (broad) topics, we thus obtained intra-topic plagiarism and inter-topic plagiarism.

4 Corpus Validation

This section reports on validation results about the "quality" of the plagiarism cases created for our corpus. We compare both artificial plagiarism cases and simulated plagiarism cases to cases of the two corpora Clough09 and METER. Presuming that the authors of these corpora put their best efforts into case construction and annotation, the comparison gives insights whether our scale-up strategies are reasonable in terms of case quality. To foreclose the results, we observe that simulated plagiarism and, in particular, artificial plagiarism behave similar to the two handmade corpora. In the light of the employed strategies to construct plagiarism this result may or may not be surprising—however, we argue that it is necessary to run such a comparison in order to provide a broadly accepted evaluation framework in this sensitive area.

The experimental setup is as follows: given a plagiarism case $s = \langle s_{\text{plg}}, d_{\text{plg}}, s_{\text{src}}, d_{\text{src}} \rangle$, the plagiarized passage s_{plg} is compared to the source passage s_{src} using 10 different retrieval models. Each model is an n-gram vector space model (VSM) where n ranges from 1 to 10 words, employing stemming, stop word removal, tf-weighting, and the cosine similarity. Similarity values are computed for all cases found in each corpus, but since the corpora are of different sizes, 100 similarities are sampled from each corpus to ensure comparability.

The rationale of this setup is as follows: a well-known fact from near-duplicate detection is that if two documents share only a few 8-grams—so-called shingles—it is highly probable that they are duplicates [5]. Another well-known fact is that two documents which are longer than a few sentences and which are exactly about the same topic will, with a high probability, share a considerable portion of their vocabulary. I.e., they have a high

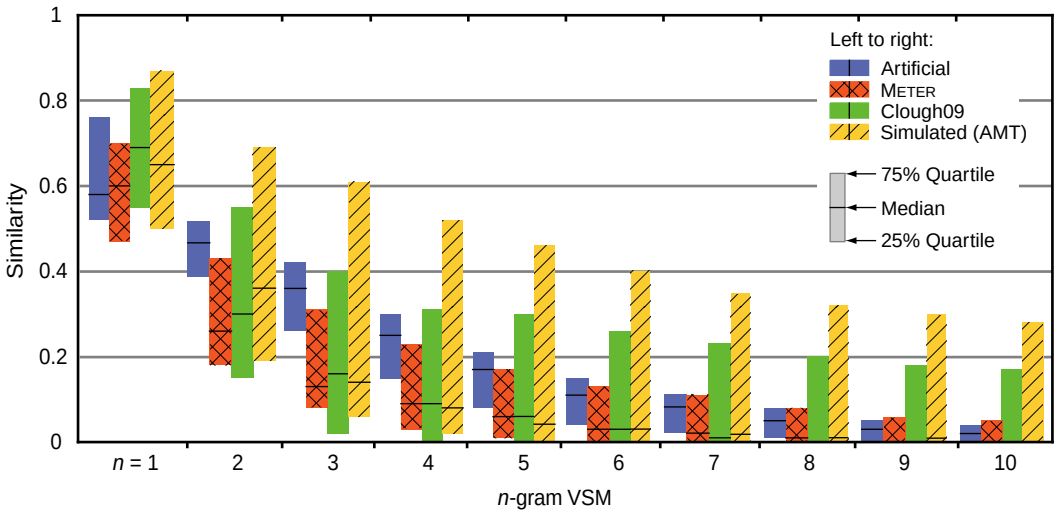

Figure 1: Comparison of four corpora of text reuse and plagiarism: each box plot shows the middle range of the measured similarities when comparing source passages to their rewritten versions. Basis is an n-gram VSM, where $n \in \{1, 2, \ldots, 10\}$ words.

similarity under a 1-gram VSM. It follows for plagiarism detection that a common shingle between s_{plg} and s_{src} pinpoints very accurately an unobfuscated portion of s_{plg}, while it is inevitable that even a highly obfuscated s_{plg} will share a portion of its vocabulary with s_{src}. The same holds for all other kinds of text reuse.

Figure 1 shows the obtained similarities, contrasting each n-gram VSM and each corpus. The box plots show the middle 50% of the respective similarity distributions as well as median similarities. The corpora divide into groups with comparable behavior: in terms of the similarity ranges covered, the artificial plagiarism compares to the METER corpus, except for $n \in \{2, 3\}$, while the simulated plagiarism from the Clough09 corpus behaves like that from our corpus, but with a different amplitude. In terms of median similarity, METER, Clough09, and our simulated plagiarism behave almost identical, while the artificial plagiarism differs. Also note that our simulated plagiarism as well as the Clough09 corpus contain some cases which are hardly obfuscated.

We interpret these results as follows: (1) Different kinds of plagiarism and text reuse do not differ very much under n-gram models. (2) Artificial plagiarism, if carefully generated, is a viable alternative to simulated plagiarism cases and real text reuse cases. (3) Our strategies to scale-up the construction of plagiarism corpora works well compared to existing, handmade corpora.

5 Summary

Current evaluation methodologies in the field of plagiarism detection research have conceptual shortcomings and allow only for a limited comparability. Our research contributes right here: we present tailored performance measures for plagiarism detection and the large-scale corpus PAN-PC-10 for the controlled evaluation of detection algorithms. The corpus features various kinds of plagiarism cases, including obfuscated cases that have been generated automatically and manually. An evaluation of the corpus in relation to previous corpora reveals a high degree of maturity. Until now, 31 plagiarism detectors have been compared using our evaluation framework. This high number of systems has been achieved based on two benchmarking workshops in which the framework was employed and developed, namely PAN'09 [15] and PAN'10 [16]. We hope that our framework will be beneficial as a challenging and yet realistic test bed for researchers in order to pinpoint the room for the development of better plagiarism detection systems.

Acknowledgements

We thank Andreas Eiselt for his devoted work on the corpus over the past two years. This work is partially funded by CONACYT-Mexico and the MICINN project TEXT-ENTERPRISE 2.0 TIN2009-13391-C04-03 (Plan I+D+i).

Bibliography

[1] Omar Alonso and Stefano Mizzaro. Can We Get Rid of TREC Assessors? Using Mechanical Turk for Relevance Assessment. In *SIGIR'09: Proceedings of the Workshop on The Future of IR Evaluation*, 2009.

[2] Vamshi Ambati, Stephan Vogel, and Jaime Carbonell. Active learning and crowd-sourcing for machine translation. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, Valletta, Malta, may 2010. European Language Resources Association (ELRA). ISBN 2-9517408-6-7.

[3] Jeff Barr and Luis Felipe Cabrera. AI Gets a Brain. *Queue*, 4(4):24–29, 2006. ISSN 1542-7730. doi: 10.1145/1142055.1142067.

[4] Regina Barzilay and Lillian Lee. Learning to Paraphrase: An Unsupervised Approach Using Multiple-Sequence Alignment. In *NAACL'03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 16–23, Morristown, NJ, USA, 2003. Association for Computational Linguistics. doi: 10.3115/1073445.1073448.

[5] Andrei Z. Broder. Identifying and Filtering Near-Duplicate Documents. In *COM'00: Proceedings of the 11th Annual Symposium on Combinatorial Pattern Matching*, pages

1–10, London, UK, 2000. Springer-Verlag. ISBN 3-540-67633-3.

[6] Manuel Cebrian, Manuel Alfonseca, and Alfonso Ortega. Towards the Validation of Plagiarism Detection Tools by Means of Grammar Evolution. *IEEE Transactions on Evolutionary Computation*, 13(3):477–485, June 2009. ISSN 1089-778X.

[7] Paul Clough. Plagiarism in Natural and Programming Languages: An Overview of Current Tools and Technologies. Internal Report CS-00-05, University of Sheffield, 2000.

[8] Paul Clough. Old and New Challenges in Automatic Plagiarism Detection. National UK Plagiarism Advisory Service, http://ir.shef.ac.uk/cloughie/papers/pas_plagiarism.pdf, 2003.

[9] Paul Clough and Mark Stevenson. Creating a Corpus of Plagiarised Academic Texts. In *Proceedings of Corpus Linguistics Conference, CL'09 (to appear)*, 2009.

[10] Paul Clough and Mark Stevenson. Developing A Corpus of Plagiarised Short Answers. *Language Resources and Evaluation: Special Issue on Plagiarism and Authorship Analysis (in press)*, 2010.

[11] Paul Clough, Robert Gaizauskas, and S. L. Piao. Building and Annotating a Corpus for the Study of Journalistic Text Reuse. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC-02)*, pages 1678–1691, 2002.

[12] Wiebe Hordijk, María L. Ponisio, and Roel Wieringa. Structured Review of Code Clone Literature. Technical Report TR-CTIT-08-33, Centre for Telematics and Information Technology, University of Twente, Enschede, 2008.

[13] Winter Mason and Duncan J. Watts. Financial Incentives and the "Performance of Crowds". In *HCOMP'09: Proceedings of the ACM SIGKDD Workshop on Human Computation*, pages 77–85, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-672-4. doi: 10.1145/1600150.1600175.

[14] Hermann Maurer, Frank Kappe, and Bilal Zaka. Plagiarism - A Survey. *Journal of Universal Computer Science*, 12(8): 1050–1084, 2006.

[15] Martin Potthast, Benno Stein, Andreas Eiselt, Alberto Barrón-Cedeño, and Paolo Rosso. Overview of the 1st International Competition on Plagiarism Detection. In Benno Stein, Paolo Rosso, Efstathios Stamatatos, Moshe Koppel, and Eneko Agirre, editors, *SEPLN 2009 Workshop on Uncovering Plagiarism, Authorship, and Social Software Misuse (PAN 09)*, pages 1–9. CEUR-WS.org, September 2009. URL `http://ceur-ws.org/Vol-502`.

[16] Martin Potthast, Benno Stein, Andreas Eiselt, Alberto Barrón-Cedeño, and Paolo Rosso. Overview of the 2nd International Benchmarking Workshop on Plagiarism Detection. In Benno Stein, Paolo Rosso, Efstathios Stamatatos, and Moshe Koppel, editors, *Proceedings of PAN at CLEF 2010: Uncovering Plagiarism, Authorship, and Social Software Misuse*, September 2010.

[17] Chanchal K. Roy and James R. Cordy. Scenario-Based Comparison of Clone Detection Techniques. In *ICPC '08: Proceedings of the 2008 The 16th IEEE International Conference on Program Comprehension*, pages 153–162, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3176-2.

[18] Chanchal K. Roy and James R. Cordy. Towards a Mutation-based Automatic Framework for Evaluating Code Clone Detection Tools. In *C3S2E '08: Proceedings of the 2008 C3S2E conference*, pages 137–140, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-101-9.

[19] Chanchal K. Roy, James R. Cordy, and Rainer Koschke. Comparison and Evaluation of Code Clone Detection Techniques and Tools: A Qualitative Approach. *Sci. Comput. Program.*, 74(7): 470–495, 2009. ISSN 0167-6423.

[20] Chanchal K. Roy and James R. Cordy. A survey on software clone detection research. Technical Report 2007-541, School of Computing, Queen's University at Kingston, Ontario, Canada, 2007.

[21] Geoffrey R. Whale. Identification of Program Similarity in Large Populations. *The Computer Journal*, 33(2):140–146, 1990. doi: 10.1093/comjnl/33.2.140.

[22] Ying Zhao, George Karypis, and Usama Fayyad. Hierarchical Clustering Algorithms for Document Datasets. *Data Min. Knowl. Discov.*, 10(2):141–168, 2005. ISSN 1384-5810. doi: 10.1007/s10618-005-0361-3.