Pamela Forner, Roberto Navigli, and Dan Tufis (Eds.):
CLEF 2013 Evaluation Labs and Workshop – Working Notes Papers, 23-26 September,
Valencia, Spain. ISBN 978-88-904810-3-1. ISSN 2038-4963. 2013.

Overview of the 5th International Competition on Plagiarism Detection

Martin Potthast,[1] Matthias Hagen,[1] Tim Gollub,[1] Martin Tippmann,[1]
Johannes Kiesel,[1] Paolo Rosso,[2] Efstathios Stamatatos,[3] and Benno Stein[1]

[1]Web Technology & Information Systems, Bauhaus-Universität Weimar, Germany
[2]Natural Language Engineering Lab, ELiRF, Universitat Politècnica de València, Spain
[3]Dept. of Information & Communication Systems Engineering, University of the Aegean, Greece

pan@webis.de http://pan.webis.de

Abstract This paper overviews 18 plagiarism detectors that have been evaluated within the fifth international competition on plagiarism detection at PAN 2013. We report on their performances for the two tasks source retrieval and text alignment of external plagiarism detection. Furthermore, we continue last year's initiative to invite software submissions instead of run submissions, and, re-evaluate this year's submissions on last year's evaluation corpora and vice versa, thus demonstrating the benefits of software submissions in terms of reproducibility.

1 Introduction

Text is reused in many ways, such as quotations, translations, paraphrases, summaries, and boilerplate text. Under the right circumstances, all of these kinds of text reuse may also be considered plagiarism [25]. A frequent research topic concerning text reuse and plagiarism is algorithms that detect them. Particularly the detection of plagiarism has received considerable attention in terms of publications over the past two decades. Our focus is on the evaluation of such algorithms with respect to their retrieval performance; since 2009 we have been organizing four annual competitions on plagiarism detection [26, 27, 29, 30] and this paper reports on the results of the fifth edition.[1]

During the first three editions of our lab we developed the first standardized evaluation framework for plagiarism detection [28]. A total of 32 teams of researchers took part in these evaluations, nine of whom more than once, and the framework has been adopted by the research community since. While evaluation frameworks should accurately emulate the real world around a given computational task in a controlled laboratory environment, most frameworks do so only to some extent, since they typically rest on design choices that affect their generalizability. This is also true for our framework, which has been shown to exert a number of shortcomings due to its semiautomatic construction that render it less realistic and sometimes lead to impractical algorithm design. As of last year, we started developments on a new, more realistic evaluation framework

[1] Some of the concepts found in this paper have been described earlier, so that, because of the inherently incremental nature of evaluations, and in order for this paper to be self-contained, we reuse text from these sources.

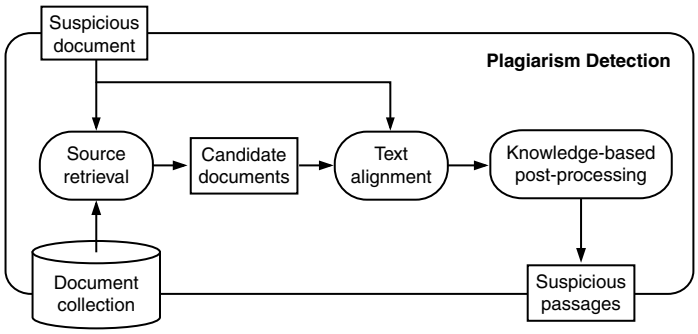

Figure 1. Generic retrieval process to detect plagiarism [40].

that consists of entirely manually generated text reuse and plagiarism [32]. This year, we employ the new framework for the second time to evaluate a total of 18 plagiarism detectors, revising it along the way.

1.1 Plagiarism Detection and its Step-wise Evaluation

Figure 1 shows a generic retrieval process to detect plagiarism in a given suspicious document d_{plg}, when also given a (very large) document collection D of potential source documents. This process is also referred to as *external* plagiarism detection since plagiarism in d_{plg} is detected by searching for text passages in D that are highly similar to text passages in d_{plg}.[2] The process is divided into three basic steps, which are typically implemented in most plagiarism detectors. First, source retrieval, which identifies a small set of candidate documents $D_{\text{src}} \subseteq D$ that are likely sources for plagiarism regarding d_{plg}. Second, text alignment, where each candidate document $d_{\text{src}} \in D_{\text{src}}$ is compared to d_{plg}, extracting all passages of text that are highly similar. Third, knowledge-based post-processing, where the extracted passage pairs are cleaned, filtered, and possibly visualized for later presentation.

1.2 Contributions

Since last year, we evaluate plagiarism detectors step-wise instead of as a whole. Our focus is on the source retrieval task and the text alignment task, and we research and develop new evaluation frameworks for each of them. In the following two sections, we detail the evaluations of both tasks. Our contributions are as follows:

1. *Software Submissions.* For the second time, we asked participants to submit their software instead of outputs of software runs. The submitted softwares were run and

[2] Another approach to detect plagiarism is called *intrinsic* plagiarism detection, where detectors are given only a suspicious document and are supposed to identify text passages in them which deviate in their style from the remainder of the document. This year, we focus on external plagiarism detection.

evaluated at our site using the TIRA experimentation platform [9]. This improves the sustainability of our evaluations because submitted softwares are maintained in executable state so that they can be run against new corpora later on.
2. *Text Alignment Evaluation Across Years.* Since software submissions were introduced last year for the text alignment task, this puts us in the position to re-evaluate last year's submissions against this year's evaluation corpora and vice versa. We report on the results of this cross-year evaluation and present a combined ranking, demonstrating the benefits of software submissions in terms or reproducibility.
3. *Survey of Retrieval Approaches.* We survey the 18 submitted softwares as described by their authors, analyze how they work, and organize them into generic retrieval processes for both of the two tasks source retrieval and text alignment.
4. *Performance Measures for Source Retrieval.* Regarding the source retrieval task, we shed light onto measuring the performance of a source retrieval algorithm. In particular, we show that near-duplicate retrieval results should be discounted when measuring source retrieval performance.

2 Source Retrieval

In source retrieval, given a suspicious document and a web search engine, the task is to retrieve all source documents from which text has been reused whilst minimizing retrieval costs. The cost-effectiveness of plagiarism detectors in this task is important since using existing search engines is perhaps the only feasible way for researchers as well as small and medium-sized businesses to implement plagiarism detection against the web, whereas search companies charge considerable fees for automatic usage.

In what follows, we describe the building blocks of our evaluation setup, provide details about the evaluation corpus and how it was constructed, discuss performance measures, survey the submitted softwares, and finally, report on the evaluation of these softwares.

2.1 Evaluation Setup

For the evaluation of source retrieval from the web, we consider the real-world scenario of an author who uses a web search engine to retrieve documents in order to reuse text from them in a document. A plagiarism detector typically uses a search engine, too, to find reused sources of a given document. Therefore, to evaluate a plagiarism detector, an evaluator must collect realistic samples of documents that contain reused text and feed them into the detector while keeping the web environment under full control. To do so in a reproducible manner, the web environment must be representative, yet static, so that evaluations of different detectors yield comparable results when done asynchronously. Meeting both constraints at the same time poses a significant engineering challenge. Over the past years, we assembled the necessary building blocks to allow for a meaningful evaluation of source retrieval algorithms; Figure 2 shows how they are connected:

- A large-scale web corpus (in our case, the ClueWeb09) that can be readily served and browsed as if it were the real web (i.e., links of delivered web pages are rewritten so that they point to the servers hosting the web corpus instead of the real web).

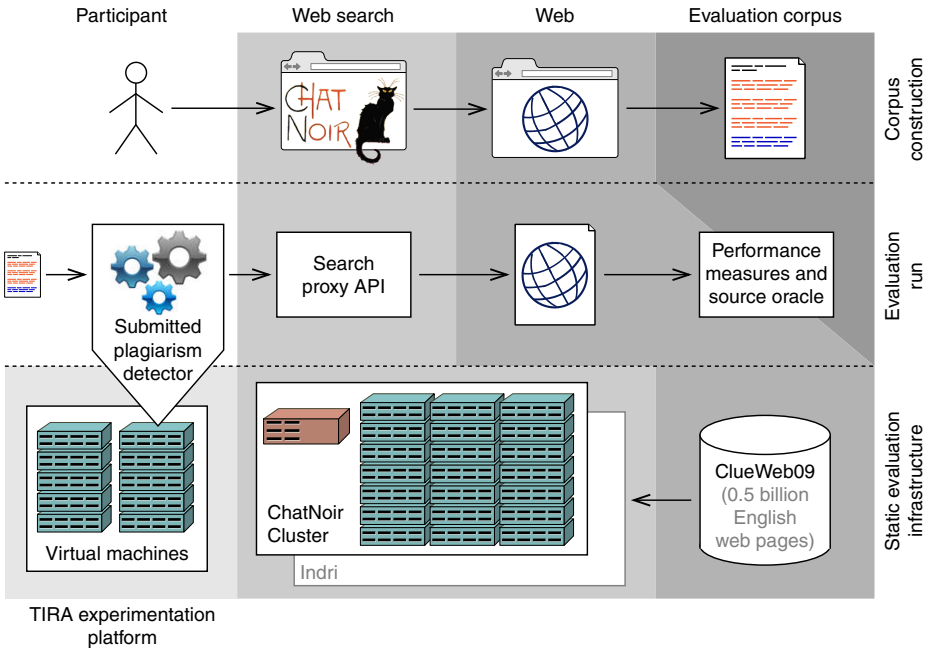

Figure 2. Overview of the building blocks used in the evaluation of plagiarism detectors that implement source retrieval. The components are organized by the two activities corpus construction and evaluation runs (top two rows). Both activities are based on a static evaluation infrastructure (bottom row) consisting of an experimentation platform, web search engines, and a web corpus.

- Web search engines (Indri and ChatNoir) that index the web corpus using two different retrieval models and that offer both a user interface as well as an API for automatic usage.
- An evaluation corpus (Webis-TRC-12) that consists of long, manually written documents from authors who searched for sources in the web corpus using one of the search engines and then reused text from sources retrieved.
- An experimentation platform (TIRA) that allows participants to submit their plagiarism detector for evaluation and for conservation in an executable state for future evaluations.
- A search proxy API that, for later analysis, monitors and logs a participant's use of the search engines as well as web page downloads from the corpus.
- Performance measures that measure the retrieval quality and cost-effectiveness of a plagiarism detector.
- A source oracle that provides feedback to a detector about whether or not a web page the detector decided to download is a true positive detection (i.e., a source document, or a duplicate thereof, that was actually used).

This setup has been used for both corpus construction as well as for evaluation runs, both of which are detailed in the following sections. Beforehand, we briefly describe

the underlying infrastructure which must be kept operational for both purposes and maintained for future evaluations.

The TIRA experimentation platform The TIRA experimentation platform is developed alongside our evaluation lab at our site [9]. It facilitates information retrieval experiments by providing a unified execution environment that allows for local instantiation of experiment software, web dissemination of evaluation results, platform independent software development, and result retrieval and visualization. TIRA itself consists of a number of building blocks the full description of which is beyond the scope of this report. Suffice it to say that one of them, depicted in Figure 2 bottom left, facilitates both platform independent software development and software submissions at the same time by its capability to create and remote control virtual machines on which our lab's participants deploy their plagiarism detectors.

The ClueWeb, Indri, and ChatNoir The currently most widely adopted web crawl that is regularly used for large-scale web search-related evaluations is the ClueWeb corpus 2009 (ClueWeb09).[3] The corpus consists of about one billion web pages, half of which are English ones. It has been successfully employed to evaluate retrieval models and search engines within the annual TREC evaluation conference[4] and has therefore been widely adopted. As of this year, an updated version of the corpus has been released,[5] however, our evaluation is still based on the 2009 version.

Indri[6] and ChatNoir [31] are currently the only publicly available search engines that index the ClueWeb09 corpus. Indri's retrieval model combines language modeling and inference networks [20], whereas ChatNoir implements the classic BM25F model [33] and incorporates PageRank and SpamRank scores. This way, two of the most widespread retrieval models are available within our evaluation setup. For developer convenience, we also provide a proxy server which unifies the APIs of the search engines. At the same time, the proxy server logs all accesses to the search engines for later performance analysis.

2.2 Evaluation Corpus

The evaluation corpus employed for source retrieval is based on the Webis text reuse corpus 2012 (Webis-TRC-2012) [32]. The corpus consists of 297 documents that have been written by 27 writers who worked with our setup as shown in the first row of Figure 2: given a topic, a writer used ChatNoir to search for source material on that topic while preparing a document of 5700 words length on average, reusing text from the found sources. The writers were instructed to modify their text reuse as much as they deemed necessary to avoid automatic detection. To do so, writers applied two general strategies, namely paraphrasing of a reused passage and interleaving of two or more reused passages from different source documents. Some writers made only superficial

[3] http://lemurproject.org/clueweb09
[4] http://trec.nist.gov
[5] http://lemurproject.org/clueweb12
[6] http://lemurproject.org/clueweb09/index.php#Services

modifications while others made many, so that a spectrum of cases ranging from no paraphrasing and no interleaving to much paraphrasing and much interleaving can be observed.

Last year, we sampled 40 documents from the Webis-TRC-2012 as training and test documents. This year, these documents were provided for training, and another 58 documents were sampled as test documents. The remainder of the corpus will be used within future labs on this task.

2.3 Performance Measures

Given a suspicious document d_{plg} that contains passages of text that have been reused from a set of source documents D_{src}, we measure the retrieval performance of a source retrieval algorithm in retrieving D_{src} in terms of precision and recall. Let D_{ret} denote the set of documents that are retrieved by a source retrieval algorithm when given d_{plg}, then it may seem straightforward to define precision as $prec = |D_{\text{ret}} \cap D_{\text{src}}|/|D_{\text{ret}}|$ and recall as $rec = |D_{\text{ret}} \cap D_{\text{src}}|/|D_{\text{src}}|$. However, this definition turns out to be overly simplistic since it disregards an important side-effect of using a large-scale web corpus for evaluation, namely near-duplicate web documents.

The web is rife with documents that are near-duplicates of each other [4]. A source retrieval algorithm may therefore retrieve a document $d_{\text{ret}} \in D_{\text{ret}}$ that is almost the same as a source document $d_{\text{src}} \in D_{\text{src}}$ in terms of its main contents, but not exactly the same. Because of this, d_{ret} would be counted falsely as a negative detection despite the fact that a human assessor would consider it a true positive one. To relax this constraint, we employ a near-duplicate detector to judge whether a retrieved document d_{ret} is a true positive detection; i.e., whether a document d_{src} exists in d_{plg}'s set of source documents D_{src} that is a near-duplicate of d_{ret}. If one of the following conditions holds, we say that d_{ret} is a true positive detection for a given pair of d_{src} and d_{plg}:

1. *Equality.* The document d_{ret} is a true positive detection of d_{src} if $d_{\text{ret}} = d_{\text{src}}$.
2. *Similarity.* The document d_{ret} is a true positive detection of d_{src} if their n-gram Jaccard similarity is above 0.8 for $n = 3$, above 0.5 for $n = 5$, and above 0 for $n = 8$. The thresholds have been determined experimentally.
3. *Containment.* The document d_{ret} is a true positive detection of d_{src} if the passages in d_{plg} known to be reused from d_{src} are contained in d_{ret}. Containment is measured as asymmetrical set overlap of the passages' set of n-grams regarding that of d_{ret}, so that the overlap is above 0.8 for $n = 3$, above 0.5 for $n = 5$, and above 0 for $n = 8$.

This way of determining whether a retrieved document d_{ret} is a true positive detection inherently entails inaccuracies, so that not all near-duplicates of a source document d_{src} will be considered true positive detections. While there is no straightforward way to solve this problem, this error source affect all detectors at the same time, still allowing for relative comparisons.

Let d_{dup} denote a near-duplicate of a given d_{src} that would be considered a true positive detection according to the above conditions if it was retrieved by a source retrieval algorithm. Note that every d_{src} may have more than one near-duplicate and

every d_{dup} may be a near-duplicate of more than one source document. Further, let D_{dup} denote the set of all near-duplicates of a given set of source documents D_{src} of d_{plg} and let D'_{ret} denote the subset of D_{src} that have at least one corresponding true positive detection in D_{ret}:

$$D_{\text{dup}} = \{d_{\text{dup}} \mid \exists d_{\text{src}} \in D_{\text{src}} : d_{\text{dup}} \text{ is a true positive detection of } d_{\text{src}}\},$$
$$D'_{\text{ret}} = \{d_{\text{src}} \mid d_{\text{src}} \in D_{\text{src}} \text{ and } \exists d_{\text{ret}} \in D_{\text{ret}} : d_{\text{ret}} \text{ is a true positive detection of } d_{\text{src}}\}.$$

Based on these sets, we define precision and recall of D_{ret} regarding D_{src} and d_{plg} as follows:

$$\text{precision} = \frac{|D_{\text{ret}} \cap D_{\text{dup}}|}{|D_{\text{ret}}|}, \qquad \text{recall} = \frac{|D'_{\text{ret}} \cap D_{\text{src}}|}{|D_{\text{src}}|}.$$

Rationale for this definition is the fact that, retrieving more than one near-duplicate of a source document does not decrease precision, but it does not increase recall, either, since no additional information is obtained.

Finally, to measure the cost-effectiveness of a source retrieval algorithm in retrieving D_{ret}, we count the numbers of queries and downloads made and compute the workload in terms of queries and downloads until the first true positive detection is made.

The Source Oracle As described at the outset, the task subsequent to source retrieval is text alignment, where the candidate sources found are compared in closer detail with a given suspicious document (see Figure 1). Because of this connection, one problem of implementing and evaluating a source retrieval algorithm is that a working text alignment algorithm is required a priori. Since such an algorithm is not at hand, we decouple the source retrieval task from the text alignment task by means of a source oracle so as to allow for participation without the need to develop a text alignment algorithm. The oracle automatically enriches a downloaded document with information about whether or not it is considered a true positive source for the given suspicious document. Note that the oracle employs the aforementioned conditions to determine whether a document is a true positive detection. However, the oracle does not, yet, tell for which part of a suspicious document a downloaded document is a true positive detection detection. Hence, applying a custom text alignment strategy can still be beneficial.

2.4 Survey of Retrieval Approaches

Nine of the 14 participants submitted runs for the source retrieval task, eight of whom also submitted a notebook describing their approach. An analysis of these descriptions reveals the same building blocks that were commonly used in last year's source retrieval algorithms: (1) chunking, (2) keyphrase extraction, (3) query formulation, (4) search control, and (5) download filtering. Some participants simply reused their previous approach; in what follows, we describe the new or changed ideas in detail.

Chunking Given a suspicious document, it is divided into (possibly overlapping) passages of text. Each chunk of text is then processed individually. Rationale for chunking the suspicious document is to evenly distribute "attention" over a suspicious document

so that algorithms employed in subsequent steps are less susceptible to unexpected characteristics of the suspicious document.

The chunking strategies employed by the participants are no chunking (i.e., the whole document as one chunk) [3, 18, 42], 50-line chunks [3], TextTiling [14] to identify the topically related passage and therein 4-sentence chunking [13], paragraph chunking [19, 42], anomaly sections based on intrinsic plagiarism detection [21], 100-word chunks [43], 5-sentence chunks [44], and combinations thereof. Note that chunks typically are stated as non-overlapping. An interesting question could be to identify the potential of overlapping chunks (except maybe in the cases of the whole-document and TextTiling chunks). Typical plagiarism cases have no fixed length and overlapping chunks would reduce the risk of, for instance, having more than one source in one chunk of 50 lines or 100 words, etc. Furthermore, relying on the given document structure (e.g., chunking by lines or paragraphs) bears the risk of failing for some unseen documents that are not as well-formatted as the ones in our evaluation corpus.

Keyphrase Extraction Given a chunk, keyphrases are extracted from it in order to formulate queries with them. Rationale for keyphrase extraction is to select only those phrases (or words) which maximize the chance of retrieving source documents matching the suspicious document. Keyphrase extraction may also serve as a means to limit the amount of queries formulated, thus reducing the overall costs of using a search engine. This step is perhaps the most important one of a source retrieval algorithm since the decisions made here directly affect the overall performance: the fewer keywords are extracted, the better the choice must be or recall is irrevocably lost.

Phrasal search was provided by the Indri search engine. However, only Lee et al. [19] made use of Indri. Some participants use single keywords while others extract whole phrases. Most of the participants preprocessed the suspicious document by removing stop words before the actual keyphrase extraction. In particular, Elizalde [3] applies three different strategies. The first approach generates one query per 50-lines chunk containing the top-10 words scored by $tf \cdot idf$ values; a word's document frequency is obtained from the external Brown corpus. The second approach uses ten queries, each of which is formed by one of the ten longest named entities extracted from the whole document using the Python Natural Language Toolkit NLTK. The third approach uses 15 queries, each of which is formed by one of the top-15 noun phrases extracted via Barker and Cornacchia's head noun phrase extractor [1]. Haggag and El-Beltagy [13] use KPMiner [2] to extract one keyphrase per topically related passage and combine this phrase with the rarest word (frequency on document level) of each 4-sentence chunk within the passage until the query contains ten keywords. Kong et al. [18] apply two different strategies. First, they use the top-20 words scored by $tf \cdot idf$ values from the whole document; a word's document frequency is obtained from the external Wall Street Journal corpus. Second, they use a Pat Tree [10] to extract one 2-gram, one 3-gram, two 4-grams, and forty 5-grams with highest $tf \cdot idf$ scores that contain at least one of the top-10 $tf \cdot idf$ terms. Lee et al. [19] use the most unique 8-gram from each paragraph (uniqueness determined via the Google Books n-grams) that starts with a word that is not contained in any keyphrase obtained for a previous paragraph. Nourian [21] use the first ten keywords of a chunk as one phrase (stopping

or keyword extraction method not specified any further). Suchomel et al. [42] apply three different strategies. First, they use the top-5 words scored by $tf \cdot idf$ values from the whole document; a word's document frequency is obtained from an external web crawl. These top-5 keywords are then also combined with their most frequent two or three term collocations. Second they use a "representative" sentence of at least 6 words length from passages in the document that an intrinsic plagiarism detector identified as differing according to writing style; however, what makes a sentence representative is not explained. Third, for a paragraph they extract the longest sentence. Veselý et al. [43] use all the longest non-overlapping n-grams ($n \geq 5$, n is odd) that return less than 300 but more than 0 results. To determine these keyphrases they basically employ the open end query formulation [36] with a query-level version of the User-over-Ranking hypothesis [39, 11]. Williams et al. [44] use a very similar and simplistic keyphrase extraction strategy: the first three disjunct sequential 10-grams of each "reduced" chunk (only nouns, adjectives and verbs) form the keyphrases. Note that this is very similar to the winning approach from 2012, where Jayapal [15] used the first such 10-gram per chunk only (also allowing pronouns).

Altogether, the participants' approaches to keyphrase extraction can basically be divided into four different categories. (1) Rather simplistic strategies that identify keyphrases by chunking the whole document into some longer n-grams. This probably conforms with the folklore human strategy of identifying some suspicious n-gram in a suspicious document and submitting this n-gram to a search engine. Using all longer n-grams probably also "hits" parts of the n-grams a human would have chosen. Thus, it is interesting to analyze the final performance of approaches that use this kind of keyphrases (cf. Section 2.5). (2) Another very common strategy is to use the $tf \cdot idf$-wise highest scoring words or phrases. (3) Notably, this year, for the first time, two participants also use keyphrase extraction schemes obtained from the research community around this topic. (4) Some participants do not rely on one strategy alone but combine different approaches for keyphrase extraction. This way, just as with chunking, the risk of algorithm error is further diminished and it becomes possible to exploit potentially different sources of information that complement each other.

Query Formulation Given sets of keywords extracted from chunks, queries are formulated which are tailored to the API of the search engine used. Rationale for this is to adhere to restrictions imposed by the search engine and to exploit search features that go beyond basic keyword search (e.g., Indri's phrasal search). The maximum number of search terms enforced by ChatNoir is 10 keywords per query while Indri allows for longer queries. Interestingly, most of the participants hardly combine keyphrases into one query apart from merging, for instance, the top-5 $tf \cdot idf$ terms, then the next five etc. This way, most participants explicitly try to formulate non-overlapping queries (i.e., they do not use the same keyword in more than one query) except for some of the participants that basically use all the longer n-grams in the suspicious document. This non-overlap is in line with many query-by-document strategies but in contrast to previous source retrieval strategies that were shown to better identify highly related documents than non-overlapping queries [12]. Also note that none of the participants made use of advanced search operators offered by Indri or ChatNoir, such as the facet

to search for web pages of at least 300 words of text, and the facet to filter search results by readability.

Search Control Given a set of queries, the search controller schedules their submission to the search engine and directs the download of search results. Rationale for this is to dynamically adjust the search based on the results of each query, which may include dropping queries, reformulating existing ones, or formulating new ones based on the relevance feedback obtained from search results. Some teams did not implement a search controller and simply submit all formulated queries. The ones who implemented search contral applied the following ideas.

Haggag and El-Beltagy [13] drop a query when more than 60% of its terms are contained in a previous downloaded document. Lee et al. [19] stop submitting queries when most of the suspicious document is found as plagiarized or the number of queries exceeds the number of paragraphs in the suspicious document. However, it remains unclear what "most of a document" means and why paragraphs are a good upper bound on the query budget. In practice, not all documents are well-formatted. A document without paragraph breaks would then allow for a single query only. Suchomel et al. [42] schedule queries dependent on the keyphrase extractor which extracted the words: the order of precedence corresponds to the order in which they have been explained above. Whenever later queries were formulated for portions of the suspicious document that were already mapped to a source, these queries are not submitted and discarded from the list of open queries.

Note that none of the teams did try to reformulate existing queries or formulating new ones based on the available number of search results, the search snippets, or the downloaded documents, which leaves significant room for improvement.

Download Filtering Given a set of downloaded documents, a download filter removes all documents that are probably not worthwhile being compared in detail with the suspicious document. Rationale for this is to further reduce the set of candidates and to save invocations of the subsequent detailed comparison step.

In particular, Elizalde [3] focuses on the top-10 results of a query and downloads a result document when at least 90% of the words in a 160-character snippet are contained in the suspicious document. Haggag and El-Beltagy [13] only consider the top-ranked result and download it when at least 50% of the query terms are contained in a 500 character snippet. It remains unclear whether checking not single words but phrases from the snippets could be beneficial. Kong et al. [18] download a document when the cosine similarity of the snippet (presumably 500 characters long) and the suspicious document exceeds some threshold (not specified in the paper). Taking into account that ChatNoir's snippets are centered around the passage of a search result that contains most of the query's terms, the cosine similarity should typically be rather high for all search results (the query terms in the snippet are also contained in the suspicious document). Lee et al. [19] download the top-k results of a query (k is not specified) and documents that appear frequently in the results (frequency threshold not specified). Suchomel et al. [42] download documents when more than 20% of the word 2-grams in the 500 character snippet also appear in the suspicious document. Veselý et al. [43] first compute

Table 1. Source retrieval results with respect to retrieval performance and cost-effectiveness.

Team (alphabetical order)	Downloaded Sources			Total Workload		Workload to 1st Detection		No Detection	Runtime
	F_1	Precision	Recall	Queries	Downloads	Queries	Downloads		
Elizalde	0.17	0.12	0.44	44.50	107.22	16.85	15.28	5	241.7 m
Gillam	0.04	0.02	0.10	16.10	33.02	18.80	21.70	38	**15.1 m**
Haggag	0.44	**0.63**	0.38	32.04	**5.93**	8.92	**1.47**	9	152.7 m
Kong	0.01	0.01	**0.65**	48.50	5691.47	2.46	285.66	**3**	4098.0 m
Lee	0.35	0.50	0.33	44.04	11.16	7.74	1.72	15	310.5 m
Nourian	0.10	0.15	0.10	**4.91**	13.54	**2.16**	5.61	27	25.3 m
Suchomel	0.06	0.04	0.23	12.38	261.95	2.44	74.79	10	1637.9 m
Veselý	0.15	0.11	0.35	161.21	81.03	184.00	5.07	16	655.3 m
Williams	**0.47**	0.55	0.50	116.40	14.05	17.59	2.45	5	1163.0 m

queries without submitting them and submit all queries computed when 20% of the words in the suspicious document are contained in the queries without computing any new query afterwards. The retrieval scores of ChatNoir are used to compute the sum of these scores for a document over all queries and in the end the 15 documents with highest sum are downloaded (however, this is not consistent with the overall workload we measured, which is around 31 downloads per suspicious document). Williams et al. [44] download the top-3 documents whose snippets share at least five word 5-grams with the suspicious document.

2.5 Evaluation Results

Table 1 shows the performances of the nine plagiarism detectors that implemented source retrieval. Since there is currently no formula to organize retrieval performance and cost-effectiveness into an absolute order, the detectors are ordered alphabetically, whereas the best performance value for each metric is highlighted. As can be seen, there is no single detector that performs best on all accounts. Rather, different detectors have different characteristics. The detector of Williams et al. [44] achieves the best trade-off between precision and recall and therefore the best F_1 value. This detector is followed closely by that of Haggag and El-Beltagy [13], which achieves best precision but mediocre recall, whereas the detector of Kong et al. [18] achieves best recall at the cost of poor precision. It is not easy to decide which of these detectors solves the task best, since each of them may have their justification in practice. For example, the detector of Haggag and El-Beltagy downloads only about six documents on average per suspicious document and minimizes the time to first detection. Despite the excellent trade-off of Williams et al.'s detector, it incurs the second-highest costs in terms of queries on average, which is more than thrice as much as the other mentioned detectors. Kong et al.'s detector has highest download costs, but one may argue that downloads are much cheaper than queries, and that in this task recall is more important than precision.

Interestingly, the ensemble of all submitted approaches would achieve an average recall of 0.82 retrieving all sources for 23 topics. Only for eight topics the recall is

below 0.65 (which is the best individual average recall). For just three topics none of the detectors detects a source (one of which actually has none).

3 Text Alignment

In text alignment, given a pair of documents, the task is to identify all contiguous passages of reused text between them. The challenge with this task is to identify passages of text that have been obfuscated, sometimes to the extent that, apart from stop words, little lexical similarity remains between an original passage and its plagiarized counterpart. Consequently, for evaluators, the challenge is to provide a representative corpus of documents that emulate this situation. To study this task, we employ a corpus construction methodology similar to that which has been used in previous evaluations of this task, while fixing some of its deficiencies. We evaluate the performance of plagiarism detectors based on the traditionally employed measures. Finally, we exploit the benefits of software submissions for the first time and evaluate the detectors that have been submitted last year on this year's evaluation corpus and vice versa.

3.1 Evaluation Corpus

The evaluation corpus for text alignment is also based on the aforementioned Webis-TRC-13. But instead of employing the documents of that corpus directly, pairs of documents that comprise reused passages have been constructed automatically, as was done in previous years [28]. One frequent point of criticism about automatically generating plagiarism is that it is difficult to ensure that documents between which text is plagiarized are about the same topic, so that plagiarism could be simply detected by analyzing topic drift [32]. Using the documents that have been retrieved manually as sources for the documents of the Webis-TRC-13 as a basis for constructing plagiarism cases, however, allows us to mitigate this problem.

Corpus Construction The corpus is constructed within eight steps:

1. *Documents.* The documents used for our corpus are web documents obtained from the ClueWeb 2009 corpus. We have compiled a set of documents on 145 topics which have been manually searched, browsed, and found relevant to a given topic. For each topic, we collected a set D_{topic} that contains between 1 and 270 documents for a total of 10 630 documents. The documents have been judged by the writers who wrote documents on these topics for the aforementioned Webis-TRC-2012.
2. *Pre-Processing.* The HTML documents were converted to plain text, extracting their main content using the BoilerPipe library. Passages of text with eight words or less were discarded as well as documents with less than 100 words. In total, 6500 documents remained after pre-processing, divided into 144 remaining topics with at least two and up to 170 documents.
3. *Withheld Documents.* For each topic, one document is withheld in order not to be used for plagiarism.

4. *Source Set Formation.* Each of the suspicious documents that are generated in a later step has a set of source documents D_{src}. In this step, these sets are formed by randomly choosing a topic from which to draw documents, a targeted size $|D_{\mathrm{src}}|$ between 5 and 75 documents, and documents from D_{topic} until the targeted size is reached or no further documents are available: in each iteration, a document is chosen at random from D_{topic} and added to D_{src}, unless a duplicate of the chosen document is already present in D_{src}. In this connection, we consider two documents duplicates if their n-gram cosine similarity is above 0.6 for $n = 1$, above 0.25 for $n = 3$, and above 0 for $n = 8$. These thresholds were experimentally determined with respect to the documents used.[7] A total of 520 source sets were created this way.
5. *Withheld Source Sets.* In addition to the above source sets, for each topic one source set was created similarly but ensuring that no sentence of a source document has a duplicate sentence in the withheld document of that topic chosen in Step 3. Two sentences are considered duplicates if their 1-gram cosine similarity is above 0.9.
6. *Passage Extraction.* Assuming a log-normal distribution of document lengths, we estimate its parameters based on the documents withheld in Step 3. For a given source set D_{src} created in Step 4, we extract a set of passages from its documents, so that each passage is at least 50 words long, and every source document contributes at least one passage. Given these constraints, we sample the number of passages to be extracted per source document from a Poisson distribution with $\lambda = 3$, favoring lesser but larger passages. Further, passages are drawn so they are no duplicates of previously drawn passages (1-gram cosine similarity above 0.9), so they are not adjacent to previously drawn passages, and so that at least one passage is drawn per document. The resulting 520 passage sets contained between 5 and 134 passages.
7. *Obfuscation.* Every passage from the passage sets extracted in Step 6 is obfuscated in order to emulate plagiarist behavior based on the following four strategies: no obfuscation, random obfuscation, and, for the first time, cyclic translation obfuscation and summary obfuscation. With the exception of summary obfuscation which was constructed independently of the rest of the corpus, the strategies are applied uniformly distributed ensuring that only one strategy is applied in a suspicious document. Details about these strategies can be found below.
8. *Suspicious Document Generation.* In this step, a suspicious document d_{plg} is generated by randomly concatenating an (obfuscated) passage set. Special care is taken with regard to their formatting so that paragraph breaks are no easy predictor of boundaries of reused passages.

Finally, the resulting suspicious documents are paired with their respective source documents, and the withheld documents are paired with the withheld source sets from Step 5 to form examples of document pairs that do not contain plagiarism. The corpus contains in total 3653 suspicious documents and 4774 source documents, which are grouped into

[7] Note that the duplicate detector differs from that applied in source retrieval. In source retrieval, we adjusted the duplicate detector to maximize precision and therefore false positive detections, whereas in text alignment, we maximize recall for the same reason (i.e., to make sure that no unintended duplication is found between pairs of documents that may lead text alignment algorithms astray).

10000 pairs, so that there are 6000 pairs containing plagiarism (i.e., 2000 for each of the mentioned obfuscation strategies), 2000 containing unobfuscated plagiarism, and 2000 without plagiarism. Half of this corpus has been released as training corpus, and the other half was used as test corpus.

Random Obfuscation Random obfuscation is a naïve approach to obfuscation in that the resulting passages are not human-readable and bear no semantics. The purpose of this type of obfuscation is to test whether text alignment algorithms are capable of identifying reused passages from a bag-of-words model point of view. The obfuscation strategy itself is a sequence of random text operations such as shuffling, adding, deleting, and replacing words or short phrases at random. Replacing words is done based on a synonym database such as WordNet, and phrases are shuffled while maintaining the original part-of-speech sequence. Moreover, some sentences may be shuffled randomly. The longer the sequence of random operations, the more an obfuscated passage differs from its original, and presumably the more difficult it is to identify them automatically. This kind of obfuscation has been used in all previous evaluation corpora that have been employed to evaluate this task.

Cyclic Translation Obfuscation A new kind of obfuscation strategy we introduce this year is cyclic translation obfuscation. Here, a plagiarized passage of text is run through a sequence of translations, so that the output of one translation forms the input of the next one while the last language of the sequence is the same as the passage's original language. Rationale of this strategy is to exploit the fact that translating a text inherently involves paraphrasing it, so that translating a text back and forth between languages is a way ob obtaining alternative versions of a text without changing its semantics.

We employ the APIs of three different translation web services for this task, namely Google Translate,[8] Microsoft Translator,[9] and MyMemory.[10] In every cyclic translation sequence, all three services are employed, since preliminary experiments revealed that employing only one of the services yield little to no difference of the obfuscated text to its unobfuscated counterpart. An explanation for this may be found in the fact that the language models used for translation appear to deterministically favor one alternative translation over others, even across many languages, while different translation services are based on independently trained models which introduce more variation in a cyclic translation sequence.

A cyclic translation sequence is constructed randomly. The intermediate languages of a sequence—start and end are always English—are drawn at random from two sets of languages, namely the Indo-European languages French, German, Italian, Spanish, and Swedish, and a mixture of different language families such as Arabic, Chinese, Hebrew, Hindi, and Japanese. First, one of the two sets is chosen and then up to three intermediate languages are employed as intermediate languages.

Summary Obfuscation Another new kind of obfuscation strategy we introduce this year is summary obfuscation. Its rationale is that including an unattributed summary

[8] http://translate.google.com
[9] http://www.microsoft.com/translator
[10] http://mymemory.translated.net

of another's document in one's own text can be considered a case of plagiarism since the main ideas of the document are maintained in condensed form. Presuming that the summary is not based on a simple concatenation of some sentences from the original document, the lexical and syntactic similarities between summary and original document may be very restricted. Actually, summary obfuscation can be viewed as a form of plagiarism of ideas rather than the simple case of reusing exact phrases or sentences.

To build a corpus of plagiarism cases based on this idea, we used existing resources from the research field of automatic text summarization. In more detail, the set of original documents was taken from the Document Understanding Conference (DUC) 2001 corpus for text summarization.[11] These are newswire stories and newspaper articles originally published in Wall Street Journal, Associated Press, San Jose Mercury News, Financial times, LA Times, and FBIS. For each such document there are two summaries of approximately 100 words created by human assessors. To produce plagiarism cases, these summaries were planted into documents of the DUC 2006 text summarization corpus.[12] This corpus also comprises newswire stories and newspaper articles originally published in Associated Press, New York Times, and Xinhua News Agency. Hence, for each original document, we produced two plagiarism cases based on summary obfuscation. In addition, for each original document, eight more documents from DUC 2006 corpus were used as suspicious documents.

Given the genre of the DUC 2001 and DUC 2006 corpora, the similarity between an original document and its summary may be easily identified by using named-entity occurrences. That is, both the original document and its summary will talk about the same persons, locations, organizations, etc. To weaken this kind of similarity, we introduced some noise in the DUC 2006 documents by replacing some of their named-entities with the named-entities of the original document or the named-entities of the summaries. In particular, we randomly selected some parts of the suspicious documents of similar length to the summaries (e.g., 100 words) and transformed them to noisy areas by replacing their named-entities with those of the original text. That way, a suspicious document appears similar to the original since it refers to the same proper names, locations, organizations, etc. The popular Stanford Named Entity Recognizer[13] was used to detect time, location, organization, person, money, percent, and date entities in original and suspicious documents. In total, there are 237 original documents and 2607 suspicious ones in this part of our corpus. 474 of the suspicious documents contain plagiarized summaries, and 1896 are noisy documents.

3.2 Performance Measures

To assess the performance of the submitted detailed comparison approaches, we employ the performance measures used in previous evaluations. For this paper to be self-contained, we summarize the definition found in [28]: let S denote the set of plagiarism cases in the corpus, and let R denote the set of detections reported by a plagiarism detector for the suspicious documents. To simplify notation, a plagiarism case

[11] http://www-nlpir.nist.gov/projects/duc/data/2001_data.html
[12] http://www-nlpir.nist.gov/projects/duc/data/2006_data.html
[13] http://nlp.stanford.edu/software/CRF-NER.shtml

$s = \langle s_{\text{plg}}, d_{\text{plg}}, s_{\text{src}}, d_{\text{src}} \rangle$, $s \in S$, is represented as a set $\mathbf{s}$ of references to the characters of d_{plg} and d_{src}, specifying the passages s_{plg} and s_{src}. Likewise, a plagiarism detection $r \in R$ is represented as $\mathbf{r}$. Based on this notation, precision and recall of R under S can be measured as follows:

$$prec(S, R) = \frac{1}{|R|} \sum_{r \in R} \frac{|\bigcup_{s \in S}(\mathbf{s} \sqcap \mathbf{r})|}{|\mathbf{r}|}, \qquad rec(S, R) = \frac{1}{|S|} \sum_{s \in S} \frac{|\bigcup_{r \in R}(\mathbf{s} \sqcap \mathbf{r})|}{|\mathbf{s}|},$$

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

Observe that neither precision nor recall account for the fact that plagiarism detectors sometimes report overlapping or multiple detections for a single plagiarism case. This is undesirable, and to address this deficit also a detector's granularity is quantified as follows:

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

where $S_R \subseteq S$ are cases detected by detections in R, and $R_s \subseteq R$ are detections of s; i.e., $S_R = \{s \mid s \in S \wedge \exists r \in R : r \text{ detects } s\}$ and $R_s = \{r \mid r \in R \wedge r \text{ detects } s\}$. Note further that the above three measures alone do not allow for a unique ranking among detection approaches. Therefore, the measures are combined into a single overall score as follows:

$$plagdet(S, R) = \frac{F_1}{\log_2(1 + gran(S, R))},$$

where F_1 is the equally weighted harmonic mean of precision and recall.

3.3 Survey of Text Alignment Approaches

Nine of the 18 submitted detectors implement text alignment, and for six of them also a notebook describing their approach has been submitted. An analysis of these notebooks reveals a number of building blocks that are commonly used to build text alignment algorithms: (1) seeding, (2) extension, and (3) filtering. Text alignment is closely related to gene sequence alignment in bioinformatics, of which the terminology is borrowed: all of this year's approaches to text alignment implement the so-called seed and extend-paradigm which is frequently applied in gene sequence alignment. In what follows, we describe them in detail.

Seeding Given a suspicious document and a source document, matches (so-called „seeds") between the two documents are identified using some seed heuristic. Seed heuristics either identify exact matches or *create* matches by changing the underlying texts in a domain-specific or linguistically motivated way. Rationale for this is to pinpoint substrings that altogether make up for the perceived similarity between suspicious and source document. By coming up with as many reasonable seeds as possible, the subsequent step of extending them into aligned passages of text becomes a lot easier.

A number of seed heuristics have been applied by this year's participants: Rodríguez Torrejón and Martín Ramos [34] use sorted word 3-grams and two kinds of sorted word

1-skip-3-grams. Kong et al. [18] use sentence pairs as seeds which exceed a given similarity threshold. Suchomel et al. [42] use sorted word 4-grams and unsorted stop word 8-grams (the latter having been introduced in [38]). Shrestha and Solorio [37] also use stop word 8-grams in addition to named entity 5-grams (i.e., word 5-grams that contain at least one named entity) as well as all other word 5-grams. The latter, however, are processed separately from the former, since otherwise they would subsume the named entitiy n-grams. Also, at the expense of runtime, they introduce inexact n-gram matching (i.e., the n-grams need not overlap entirely but their Jaccard similarity must be above a given threshold). Palkovskii and Belov [24] use sorted word 5-grams. Before computing seeds, some participants choose to collapse whitespace, reduce cases, remove stop words, and stem the remaining words, if applicable to their respective seed heuristics.

Extension Given seed matches identified between a suspicious document and a source document, they are extended into aligned text passages between the two documents of maximal length, which are then reported as plagiarism detections. Rationale for merging seed matches is to determine whether a document contains plagiarized passages at all rather than just seeds matching by chance, and to identify a plagiarized passage as a whole rather than only its fragments.

Most of the participants' extension heuristics are rule-based, merging seeds into aligned passages if they are adjacent in both suspicious and source document and the size of the gap between them is below some threshold. The exact rule depends on the seeds used, and instead of using just one rule, many participants develop sets of constraints that have to be fulfilled by aligned passages in order to be reported as plagiarism detections. Since the rules are usually highly involved with their respective setup, we exemplify only one rule set here in order to give an idea of what they may look like: Suchomel et al. [42] employ a 2-step merge heuristic, where in the first step, adjacent seed matches that are no more than 4000 chars apart are merged. The resulting passages from the first step are then merged again, considering pairs of adjacent passages in turn, and checking if the gap between them contains at least four seeds so that there is at least one seed per 10 000 chars of gap length between them. To be merged, adjacent passages further have to fulfill the constraints that their gap is smaller than 30 000 chars, that their combined size is bigger than twice the gap size, and that the ratio of seeds per chars of the adjacent passages does not drop by a factor of more than three in the potentially merged passage. The only participants who go beyond rule-based merging are Palkovskii and Belov [24], who employ clustering for unsupervised merging.

Filtering Given a set of aligned passages, a passage filter removes all aligned passages that do not meet certain criteria. Rationale for this is mainly to deal with overlapping passages and to discard extremely short passages.

Kong et al. [17] discard passages whose word overlap under a modified Jaccard coefficient is below a threshold. Suchomel et al. [41] discard overlapping passages that are shorter than 300 chars, and keep only the passages longer than 300 chars. Palkovskii and Belov [23] discard passages shorter than 190 chars. Gillam et al. [8] discard passages shorter than 50 words that have less than 0.75 cosine similarity under a vector space model. Other participants do not apply passage filtering.

Table 2. Text alignment results with retrieval performance and runtime.

Team	PlagDet	Recall	Precision	Granularity	Runtime
R. Torrejón	0.82220	0.76190	0.89484	1.00141	1.2 m
Kong	0.81896	0.81344	0.82859	1.00336	6.1 m
Suchomel	0.74482	0.76593	0.72514	1.00028	28.0 m
Saremi	0.69913	0.77123	0.86509	1.24450	446.0 m
Shrestha	0.69551	0.73814	0.87461	1.22084	684.5 m
Palkovskii	0.61523	0.53561	0.81699	1.07295	6.5 m
Nourian	0.57716	0.43381	0.94707	1.04343	40.1 m
Baseline	0.42191	0.34223	0.92939	1.27473	30.5 m
Gillam	0.40059	0.25890	0.88487	1.00000	21.3 m
Jayapal	0.27081	0.38187	0.87901	2.90698	4.8 m

Remarks Since six of this year's participants took part in previous years as well, many of them simply reuse their earlier solutions. While there is no problem with doing so, innovative ideas become less frequent compared to parameter tuning and small adjustments to an algorithm. However, this year's best performing approach submitted by Rodríguez Torrejón and Martín Ramos [34], has been evaluated for the fourth time in a row, showing that persistent development may eventually yield good results. A number of new ideas could be observed:

- Palkovskii and Belov [24] continue their development of obfuscation-specific detection approaches by targeting summary obfuscation in particular.
- Suchomel et al. [42], Rodríguez Torrejón and Martín Ramos [34], and Shrestha and Solorio [37] employ more than one seed heuristic at the same time. In particular, the seed heuristic of Stamatatos [38] based on stop word n-grams is used more often.
- Shrestha and Solorio [37] employ inexact seed matching (i.e., in order for a pair of seeds to be linked across documents, they need not be exactly equal but only approximately equal according to some similarity measure). This approach may lead to more relaxed seeding heuristics, but also introduces runtime overhead.

3.4 Evaluation Results

In this section, we report on the evaluation of this year's submissions on the aforementioned evaluation corpus. Moreover, we conduct the first cross-year evaluation of all softwares submitted last year and this year on the evaluation corpora of both years. We further differentiate performance with regard to obfuscation strategies to provide insights into how the softwares deal with different strengths of obfuscation. In addition to that, we reveal how the changes made to softwares that have been submitted in different versions in both years affect performance, and whether or not they improved. We also shed light on the question of corpus difficulty and find that last year's evaluation corpus was more difficult than this year's corpus.

Overall Results of 2013 Table 2 shows the overall performances of the nine plagiarism detectors that implement text alignment and were submitted this year. The overall best performing approach is that of Rodríguez Torrejón and Martín Ramos [34], closely

followed by that of Kong et al. [18]. The former detector has unbalanced precision and recall, while the latter does, but with worse granularity. The three new approaches submitted this year from Saremi and Yaghmaee [35], Shrestha and Solorio [37] and Nourian [21] achieve mid-range performances. Two detectors' performances do not exceed the baseline. In terms of precision and granularity, almost all detectors perform well, whereas recall sets them apart. In terms of runtime, all detectors are in the range of minutes, one requiring only 1.2 minutes, while two others lag far behind because they employ resource-intensive named entity recognition algorithms.

Cross-Year Evaluation of 2012 and 2013 Tables 3 to 6 show the performances of all 18 plagiarism detectors submitted last year and this year that implement text alignment on both years' respective evaluation corpora. The overall performance of the detectors with regard to the $plagdet$ score can be found in Table 3. As can be seen, the best performing detectors across both years are those of Oberreuter et al. [22] and Kong et al. [17], both of which have been first evaluated in 2012. This year's best performing detectors from Rodríguez Torrejón and Martín Ramos [34] comes close to them, however, only when evaluated on the 2013 evaluation corpus. On the 2012 corpus it is far off, which suggest this detector may be overfitted to the 2013 corpus.

Regarding different obfuscation strategies, it appears the detectors' performances on the 2012 corpus correlate mostly with their overall performance, but on the 2013 corpus this is not the case. Especially for summary obfuscation, the two best performing detectors are from Suchomel et al. [41, 42] which otherwise achieve mid-range performance only. The detector of Oberreuter et al. [22] fails on summarized plagiarism, while that of Kong et al. [17] achieves third-best performance on this kind of obfuscation. It is unfortunate that the detector of Palkovskii and Belov [24], which implements a detection approach that targets summary obfuscation, does not compete with the others. Regarding unobfuscated plagiarism (column "None" in the tables), it is interesting to observe that many detectors detect this kind of plagiarism with scores above 0.9 on the 2012 corpus but less so in the 2013 corpus where the scores are mostly below that number. It is unclear why this is the case, since unobfuscated plagiarism is not changed when inserted into a suspicious document with the exception of text formatting.

Table 4 shows the detectors' performances with regard to precision. In general, achieving a high precision appears to be less of a problem compared to achieving a high recall. This is underpinned by the fact that our basic baseline approach outperforms almost all detectors in precision. However, the detectors that perform best in precision typically have deficiencies in terms of recall, but not the other way around: the aforementioned overall best performing detectors achieve mid-range precision. The only obfuscation strategy that poses a comparably higher challenge in terms of precision is random high obfuscation, which has been adjusted to emulate extreme obfuscation. Table 5 shows the detectors' performances with regard to recall. The best performing detectors are the two versions submitted by Kong et al. [17, 18]. They dominate all others in terms of recall, but not each other; the 2013 version performs best on the 2012 corpus and vice versa. By contrast, this year's best performing detector by Rodríguez Torrejón and Martín Ramos [34] achieves only mid-range recall. The best performing approaches with regard to precision from Gillam et al. [8, 7] performs poor with regard to recall, suggesting that the implemented approach is too conservative for this

Table 3. Cross-year evaluation of text alignment software submissions for 2012 and 2013 with respect to $plagdet$. The darker a cell, the better the performance compared to the entire column.

Software Submission		**Obfuscation Strategies of the 2012 Evaluation Corpus**					**Entire Corpus**
Team	Year	None	Random low	Random high	Translation	Man. paraphrase	
Oberreuter	2012	0.92552	0.84416	0.40671	0.78128	0.71728	0.74575
Kong	2012	0.89899	0.82258	0.39652	0.77121	0.75884	0.73846
Kong	2013	0.87815	0.81645	0.39764	0.77576	0.73973	0.72576
Suchomel	2013	0.88169	0.81143	0.33546	0.71113	0.64713	0.68542
R. Torrejón	2012	0.93248	0.71393	0.12761	0.69631	0.67367	0.67078
R. Torrejón	2013	0.94516	0.71251	0.14697	0.73955	0.68047	0.66816
Suchomel	2012	0.93875	0.80181	0.15376	0.63538	0.61104	0.66532
Palkovskii	2012	0.82442	0.76691	0.31273	0.73679	0.62232	0.64630
Nourian	2013	0.86535	0.51824	0.06977	0.55013	0.49950	0.53270
Kueppers	2012	0.80950	0.26609	0.02645	0.48362	0.29958	0.40494
Palkovskii	2013	0.54280	0.33422	0.10561	0.46256	0.42080	0.38150
Gillam	2012	0.92933	0.04741	0.00917	0.00050	0.12409	0.31109
Gillam	2013	0.92736	0.04735	0.00917	0.00050	0.12253	0.31034
Sánchez-Vega	2012	0.60305	0.25388	0.04202	0.39758	0.26400	0.30857
Baseline		0.87712	0.06382	0.00023	0.04440	0.06137	0.20210
Jayapal	2013	0.11483	0.06265	0.01075	0.07336	0.02950	0.05753
Jayapal	2012	0.10272	0.05059	0.01349	0.05211	0.04394	0.05085

Software Submission		**Obfuscation Strategies of the 2013 Evaluation Corpus**				**Entire Corpus**
Team	Year	None	Random	Cyclic translation	Summary	
Kong	2012	0.87249	0.83242	0.85212	0.43635	0.83679
Oberreuter	2012	0.94170	0.74955	0.84618	0.13208	0.82678
R. Torrejón	2013	0.92586	0.74711	0.85113	0.34131	0.82220
Kong	2013	0.82740	0.82281	0.85181	0.43399	0.81896
Palkovskii	2012	0.88161	0.79692	0.74032	0.27507	0.79155
R. Torrejón	2012	0.88222	0.70151	0.80112	0.44184	0.78767
Suchomel	2013	0.81761	0.75276	0.67544	0.61011	0.74482
Suchomel	2012	0.89848	0.65213	0.63088	0.50087	0.73224
Saremi	2013	0.84963	0.65668	0.70903	0.11116	0.69913
Shrestha	2013	0.89369	0.66714	0.62719	0.11860	0.69551
Kueppers	2012	0.81977	0.51602	0.56932	0.13848	0.62772
Palkovskii	2013	0.82431	0.49959	0.60694	0.09943	0.61523
Nourian	2013	0.90136	0.35076	0.43864	0.11535	0.57716
Sánchez-Vega	2012	0.52179	0.45598	0.44323	0.28807	0.45923
Baseline		0.93404	0.07123	0.10630	0.04462	0.42191
Gillam	2012	0.87655	0.04723	0.01225	0.00218	0.41373
Gillam	2013	0.85884	0.04191	0.01224	0.00218	0.40059
Jayapal	2013	0.38780	0.18148	0.18181	0.05940	0.27081
Jayapal	2012	0.34758	0.12049	0.10504	0.04541	0.20169

Table 4. Cross-year evaluation of text alignment software submissions for 2012 and 2013 with respect to precision. The darker a cell, the better the performance compared to the entire column.

Software Submission		**Obfuscation Strategies of the 2012 Evaluation Corpus**					**Entire Corpus**
Team	Year	None	Random low	Random high	Translation	Man. paraphrase	
Gillam	2012	0.92606	0.91764	0.73913	0.99999	0.91521	0.89843
Gillam	2013	0.92156	0.91764	0.77273	0.99999	0.91525	0.89041
Suchomel	2012	0.88525	0.94855	0.79336	0.84107	0.91570	0.87214
Baseline		0.79534	0.88055	0.04444	0.85048	0.99794	0.86793
Oberreuter	2012	0.86141	0.95761	0.87900	0.83994	0.90922	0.86458
R. Torrejón	2013	0.91286	0.96042	0.54365	0.84533	0.97519	0.84503
R. Torrejón	2012	0.89876	0.89604	0.76443	0.83487	0.81161	0.82722
Kong	2012	0.83488	0.92876	0.75043	0.82079	0.90062	0.82389
Nourian	2013	0.79599	0.89959	0.46519	0.89704	0.94958	0.81174
Kong	2013	0.80075	0.91286	0.71982	0.81206	0.86175	0.79273
Kueppers	2012	0.81703	0.93022	0.27662	0.84225	0.96319	0.78973
Suchomel	2013	0.79102	0.82890	0.71409	0.78851	0.76546	0.74912
Palkovskii	2012	0.71048	0.86228	0.68476	0.83246	0.72824	0.69522
Jayapal	2012	0.98878	0.53703	0.24240	0.74551	0.48731	0.67591
Palkovskii	2013	0.38351	0.88968	0.70396	0.82905	0.64334	0.58447
Jayapal	2013	0.78382	0.44834	0.17098	0.72738	0.26708	0.56798
Sánchez-Vega	2012	0.59216	0.57682	0.18661	0.83989	0.84231	0.53753

Software Submission		**Obfuscation Strategies of the 2013 Evaluation Corpus**				**Entire Corpus**
Team	Year	None	Random	Cyclic translation	Summary	
Nourian	2013	0.92921	0.96274	0.95856	0.99972	0.94707
Jayapal	2012	0.98542	0.95984	0.89590	0.83259	0.94507
Baseline		0.88741	0.98101	0.97825	0.91147	0.92939
R. Torrejón	2013	0.90060	0.90996	0.89514	0.90750	0.89484
Oberreuter	2012	0.89037	0.87921	0.90328	0.98983	0.89443
Gillam	2012	0.88128	0.95572	0.97273	0.99591	0.88532
Gillam	2013	0.88088	0.95968	0.97273	0.99591	0.88487
Jayapal	2013	0.91989	0.92314	0.85653	0.68832	0.87901
Shrestha	2013	0.80933	0.92335	0.88008	0.90455	0.87461
Kueppers	2012	0.83258	0.89889	0.89985	0.86239	0.86923
Saremi	2013	0.82676	0.91810	0.84819	0.94600	0.86509
Kong	2012	0.80786	0.89367	0.85423	0.96399	0.85297
Suchomel	2012	0.81678	0.87581	0.85151	0.87478	0.84437
Kong	2013	0.76077	0.86224	0.85744	0.96384	0.82859
R. Torrejón	2012	0.81313	0.83881	0.81159	0.92666	0.82540
Palkovskii	2012	0.79219	0.84844	0.83218	0.94736	0.82371
Palkovskii	2013	0.79971	0.93137	0.82207	0.67604	0.81699
Suchomel	2013	0.69323	0.82973	0.68494	0.67088	0.72514
Sánchez-Vega	2012	0.40340	0.49524	0.37300	0.45184	0.39857

Table 5. Cross-year evaluation of text alignment software submissions for 2012 and 2013 with respect to recall. The darker a cell, the better the performance compared to the entire column.

Software Submission		**Obfuscation Strategies of the 2012 Evaluation Corpus**					**Entire Corpus**
Team	Year	None	Random low	Random high	Translation	Man. paraphrase	
Kong	2013	0.97212	0.77942	0.28378	0.74257	0.65001	0.67971
Kong	2012	0.97376	0.77716	0.27710	0.72727	0.65768	0.67877
Oberreuter	2012	0.99994	0.78029	0.26646	0.73028	0.59226	0.66072
Suchomel	2013	0.99582	0.79682	0.21922	0.64758	0.56048	0.63205
Palkovskii	2012	0.99897	0.75036	0.20687	0.66084	0.54329	0.61864
R. Torrejón	2013	0.97983	0.61697	0.08888	0.65729	0.52734	0.56609
R. Torrejón	2012	0.96883	0.59335	0.06961	0.59719	0.57758	0.56468
Suchomel	2012	0.99912	0.69438	0.08513	0.51053	0.45850	0.53778
Palkovskii	2013	0.96086	0.51128	0.08881	0.49045	0.44202	0.49151
Nourian	2013	0.95057	0.47997	0.04437	0.39671	0.36408	0.43694
Sánchez-Vega	2012	0.84717	0.28239	0.03033	0.43377	0.24522	0.34798
Kueppers	2012	0.93472	0.19246	0.01389	0.43943	0.24407	0.34535
Baseline		0.99888	0.04571	0.00011	0.03894	0.08176	0.21593
Gillam	2012	0.93503	0.02785	0.00709	0.00025	0.06656	0.19210
Gillam	2013	0.93562	0.02782	0.00709	0.00025	0.06566	0.19190
Jayapal	2013	0.55421	0.10837	0.00998	0.10898	0.05298	0.15126
Jayapal	2012	0.25459	0.06208	0.01148	0.05322	0.04942	0.08162

Software Submission		**Obfuscation Strategies of the 2013 Evaluation Corpus**				**Entire Corpus**
Team	Year	None	Random	Cyclic translation	Summary	
Kong	2012	0.94836	0.77903	0.85003	0.29892	0.82449
Kong	2013	0.90682	0.78682	0.84626	0.30017	0.81344
Saremi	2013	0.95416	0.68877	0.80473	0.10209	0.77123
Oberreuter	2012	0.99932	0.65322	0.79587	0.07076	0.76864
Suchomel	2013	0.99637	0.68886	0.66621	0.56296	0.76593
R. Torrejón	2013	0.95256	0.63370	0.81124	0.21593	0.76190
Palkovskii	2012	0.99379	0.75130	0.66672	0.16089	0.76181
R. Torrejón	2012	0.96414	0.60283	0.79092	0.29007	0.75324
Shrestha	2013	0.99902	0.71461	0.63618	0.09897	0.73814
Suchomel	2012	0.99835	0.51946	0.50106	0.35305	0.64667
Sánchez-Vega	2012	0.74452	0.43502	0.58133	0.22161	0.56225
Palkovskii	2013	0.85048	0.36420	0.49667	0.08082	0.53561
Kueppers	2012	0.83854	0.36865	0.42427	0.09265	0.51074
Nourian	2013	0.87626	0.23609	0.28568	0.07622	0.43381
Jayapal	2013	0.86040	0.18182	0.19411	0.07236	0.38187
Baseline		0.99960	0.04181	0.08804	0.03649	0.34223
Gillam	2012	0.87187	0.02422	0.00616	0.00109	0.26994
Gillam	2013	0.83788	0.02142	0.00616	0.00109	0.25890
Jayapal	2012	0.51885	0.11148	0.09195	0.04574	0.22287

Table 6. Cross-year evaluation of text alignment software submissions for 2012 and 2013 with respect to granularity. The darker a cell, the better the performance compared to the entire column.

Software Submission		**Obfuscation Strategies of the 2012 Evaluation Corpus**					**Entire Corpus**
Team	Year	None	Random low	Random high	Translation	Man. paraphrase	
Suchomel	2012	1.00000	1.00000	1.00000	1.00000	1.00000	1.00000
Suchomel	2013	1.00000	1.00190	1.00000	1.00000	1.00000	1.00040
R. Torrejón	2012	1.00000	1.00000	1.00000	1.00000	1.00249	1.00084
Oberreuter	2012	1.00000	1.02602	1.00763	1.00000	1.00000	1.00610
Kong	2012	1.00000	1.04024	1.02899	1.00000	1.00251	1.01105
Kong	2013	1.00000	1.04192	1.03318	1.00000	1.00248	1.01172
Palkovskii	2012	1.01005	1.06526	1.02232	1.00000	1.00000	1.01809
R. Torrejón	2013	1.00000	1.07692	1.05556	1.00000	1.00828	1.02050
Gillam	2013	1.00177	1.20455	1.88889	1.00000	1.00000	1.02429
Gillam	2012	1.00178	1.20455	1.88889	1.00000	1.00000	1.02436
Nourian	2013	1.00175	1.30997	1.23636	1.00000	1.07590	1.09426
Kueppers	2012	1.10980	1.29514	1.00000	1.28818	1.46228	1.27635
Sánchez-Vega	2012	1.22826	1.81558	1.36471	1.71117	1.71099	1.58308
Palkovskii	2013	1.01386	2.84501	1.81579	1.51816	1.37062	1.63842
Baseline		1.01340	1.57021	1.00000	2.19824	4.51313	2.27432
Jayapal	2012	14.36813	3.59490	2.08333	2.74925	3.11826	6.28323
Jayapal	2013	49.36929	5.89810	2.37129	4.99491	6.98626	16.78476

Software Submission		**Obfuscation Strategies of the 2013 Evaluation Corpus**				**Entire Corpus**
Team	Year	None	Random	Cyclic translation	Summary	
Gillam	2012	1.00000	1.00000	1.00000	1.00000	1.00000
Gillam	2013	1.00000	1.00000	1.00000	1.00000	1.00000
Oberreuter	2012	1.00000	1.00000	1.00000	1.00000	1.00000
Palkovskii	2012	1.00000	1.00000	1.00000	1.00000	1.00000
R. Torrejón	2012	1.00000	1.00000	1.00000	1.00000	1.00000
Suchomel	2013	1.00000	1.00000	1.00000	1.00476	1.00028
Suchomel	2012	1.00000	1.00000	1.00000	1.00610	1.00032
R. Torrejón	2013	1.00000	1.00000	1.00000	1.03086	1.00141
Kong	2012	1.00000	1.00000	1.00000	1.06452	1.00282
Kong	2013	1.00000	1.00000	1.00000	1.07742	1.00336
Sánchez-Vega	2012	1.00394	1.02200	1.03533	1.04523	1.02196
Kueppers	2012	1.02687	1.01847	1.01794	1.31061	1.03497
Nourian	2013	1.00092	1.11558	1.00485	1.34234	1.04343
Palkovskii	2013	1.00000	1.06785	1.02825	1.73596	1.07295
Shrestha	2013	1.00083	1.30962	1.26184	1.83696	1.22084
Saremi	2013	1.06007	1.29511	1.24204	2.15556	1.24450
Baseline		1.00912	1.18239	1.86726	1.97436	1.27473
Jayapal	2012	2.87916	2.15530	2.00578	2.75743	2.45403
Jayapal	2013	3.90017	2.19096	2.34218	3.60987	2.90698

task; the authors reveal their primary concerns up to now has been near-duplicate texts instead of paraphrases. In general, a visible correlation of recall performance on the entire 2012 corpus with the performances for each obfuscation strategy can be observed, and to a lesser extent on the 2013 corpus. Table 6 shows the detectors' performances with regard to granularity. Many detectors achieve perfect granularity on almost all obfuscation strategies, which may indicate that the problem of fragmented detections of a contiguous plagiarism case is under control, however, especially obfuscated plagiarism naturally poses a higher challenge with regard to this performance measure, which can be seen looking at random obfuscation as well as summary obfuscation. The only approaches that apparently do not do anything about granularity appear to be the ones of Jayapal [15, 16]. In general, however, these numbers must be taken with a grain of salt, since participants often resort to post-retrieval filtering in order to optimize granularity only for the sake of achieving a good ranking instead, while some admit that they would not do this in practice.

Comparing Detector Versions between 2012 and 2013 Since six of nine teams submitted versions of their detectors in both 2012 and 2013, this allows for an analysis of performance changes across versions of the same detector, and whether the adjustments made pay off in terms of improved performance. Such analyses are a novelty for evaluation labs such as ours and they yield insights into task design. Figure 3 shows the performance differences of each of the six detector pairs when subtracting their respective 2012 performance values from their 2013 ones for each of our four performance measures and both years' evaluation corpora. Regarding $plagdet$, half of the participants achieve a performance improvement and the other half decreased the performance of their detectors. The highest performance gain of about 0.08 $plagdet$ performance was achieved by Jayapal and Goswami [16], but only on the 2013 corpus, while Palkovskii and Belov [24] suffer a significant performance loss of at least 0.2 $plagdet$ on both corpora. Clearly, the modifications made on the latter detector should be carefully reviewed or even reverted.

Precision and recall are related measures in that one can typically be traded for the other. Regarding them, it can be seen that all but one detector decrease in precision performance, but only Jayapal and Goswami [16] and Suchomel et al. [42] materialize a return in terms of increased recall. The modifications made by Gillam [7] and Kong et al. [18] result in a slight decrease of recall, and those of Palkovskii and Belov [24] in a big loss of recall. The detector of Rodríguez Torrejón and Martín Ramos [34] improves a lot in terms of precision but only slightly in terms of recall. Finally, all but one detector suffer losses in terms of granularity. Most of these losses are rather small, except for that of Jayapal and Goswami [16] which amounts to more than 10 granularity loss.

Comparing Corpus Versions between 2012 and 2013 Software submissions not only allow for more sustainable evaluations and assessing software versions in terms of performance changes, but also for measuring the difficulty of different evaluation corpora. By evaluating every detector on the evaluation corpora of both 2012 and 2013, a distribution of comparable performance values is obtained that sheds light on how difficult it is to identify cases of plagiarism in the two corpora relative to each other. Figure 4 shows the performances of all detectors on both corpora for each of our four performance mea-

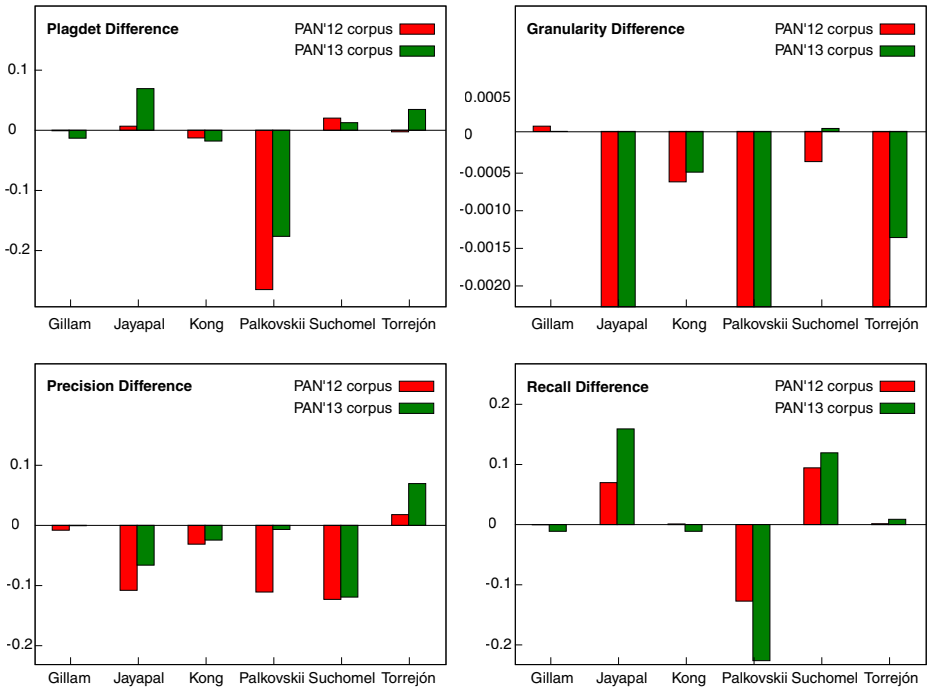

Figure 3. Performance differences of detectors of which two versions have been submitted, one last year and one this year. The differences reveal the performance changes which result from further development on these detectors both on the 2012 evaluation corpus and the 2013 corpus.

sures, ordered from high to low performance. In terms of $plagdet$, the 2013 evaluation corpus is consistently easier than the 2012 corpus. While the recall curve difference is comparable to that of the $plagdet$ curves, the precision curves are closer to each other, which indicates that precision difficulty is similar across both corpora. In terms of granularity, more than half of the detectors achieve almost equal performance. The remainder perform better on the 2013 corpus because their granularity values are smaller. As a result, our revised corpus construction process outlined above yields plagiarism cases that are more easily detected in general.

Besides these differences, the obfuscation-specific performances shown in Tables 3 to 6 show that the random obfuscation strategy employed in 2013 compares to that of random low obfuscation of 2012; despite other intentions, we did not accomplish to hit the middle ground between random low and random high obfuscation, which contributes to the 2013 corpus being less difficult. The cyclic translation obfuscation appears to be on a level of difficulty similar to that of random (low) obfuscation, since most detectors achieve similar performances on them. The most difficult portions of the 2012 corpus is random high obfuscation, and that of the 2013 corpus is summary obfuscation, which can be seen particularly when considering recall performance.

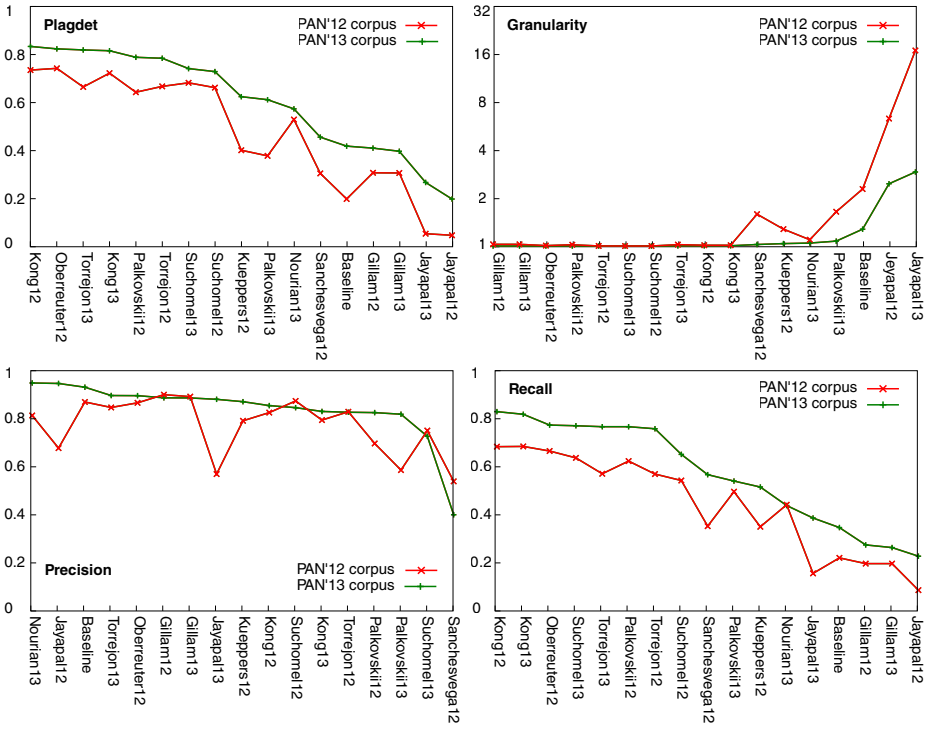

Figure 4. Performance value distribution on both the 2013 evaluation corpus and the 2012 corpus. The differences in performance across the resulting curves indicate corpus difficulty relative to each other. The bigger—smaller, in case of granularity—the area under a curve, the easier the corresponding corpus.

4 Conclusion and Outlook

With this fifth international competition on plagiarism detection at PAN 2013 we introduced a number of improvements to the evaluation methodology for plagiarism detectors. (1) By calling for software submissions instead of run submissions, we further automated the organization of evaluation labs in general. Similarly, we improved both the reproducibility and the comparability of the evaluation results. (2) We extended the evaluation setup for plagiarism source retrieval, which now is a task built on top of two search engines that index the ClueWeb corpus, a search proxy API, and a source oracle service, all of which are running on a cluster computer at our site. Moreover, by introducing performance measures that are robust against near-duplicate retrieval results we improved the task at a conceptual level. (3) For the plagiarism text alignment task we presented a new evaluation corpus that is based on manually written essays so as to further increase its realism. (4) We introduced two new forms of plagiarism obfuscation strategies, which implement new paradigms of emulating a real plagiarist's behavior

when modifying a copied passage. The two strategies are: cyclic translations, which provide for more realistic automatic paraphrasing compared to previously employed methods, and summaries, which have been obtained from a third-party data source.

This year, we collected a total of 18 plagiarism detectors from 14 teams, half of which implement source retrieval and the other half text alignment. For the task of text alignment, this is the second year in which we ask for software submissions instead of run submissions, which gives us the opportunity to conduct a cross-year evaluation of all detectors submitted in both years on all evaluation corpora available. Our cross-year evaluation reveals that this year's best performing detector cannot keep up with the best performing detectors of last year. Moreover, considering the six detectors that have been submitted in both years, we analyzed whether their retrieval performance has been improved. In fact, for three of the detectors this is not the case; i.e., the adjustments should be carefully reviewed or even reverted. Finally, we used the submitted softwares to compare the difficulty of our corpora, and we found out that this year's evaluation corpus is significantly easier than the last year's corpus in terms of detecting the contained plagiarism cases.

Altogether, we see a lot of room for further improvement with respect to both the methodology of plagiarism detection evaluation and the organization paradigm of software submissions.

Acknowledgements

We thank the participating teams of this task for their devoted work. This paper was partially supported by the WIQ-EI IRSES project (Grant No. 269180) within the FP7 Marie Curie action.

Bibliography

[1] Ken Barker and Nadia Cornacchia. Using noun phrase heads to extract document keyphrases. In Howard J. Hamilton, editor, *Advances in Artificial Intelligence, 13th Biennial Conference of the Canadian Society for Computational Studies of Intelligence, AI 2000, Montréal, Quebec, Canada, May 14-17, 2000, Proceedings*, volume 1822 of *Lecture Notes in Computer Science*, pages 40–52. Springer, 2000. ISBN 3-540-67557-4.

[2] Samhaa R. El-Beltagy and Ahmed A. Rafea. Kp-miner: A keyphrase extraction system for english and arabic documents. *Information Systems*, 34(1):132–144, 2009.

[3] Victoria Elizalde. Using Statistic and Semantic Analysis to Detect Plagiarism—Notebook for PAN at CLEF 2013. In Forner et al. [6].

[4] Dennis Fetterly, Mark Manasse, and Marc Najork. On the Evolution of Clusters of Near-Duplicate Web Pages. In *Proceedings of the 1st Latin American Web Congress, LA-WEB 2003*. IEEE, 2003. ISBN 0-7695-2058-8/03.

[5] Pamela Forner, Jussi Karlgren, and Christa Womser-Hacker, editors. *CLEF 2012 Evaluation Labs and Workshop – Working Notes Papers, 17-20 September, Rome, Italy*, 2012. ISBN 978-88-904810-3-1. URL http://www.clef-initiative.eu/publication/working-notes.

[6] Pamela Forner, Roberto Navigli, and Dan Tufis, editors. *CLEF 2013 Evaluation Labs and Workshop – Working Notes Papers, 23-26 September, Valencia, Spain*, 2013. URL http://www.clef-initiative.eu/publication/working-notes.

[7] Lee Gillam. Guess Again and See if They Line Up: Surrey's Runs at Plagiarism Detection—Notebook for PAN at CLEF 2013. In Forner et al. [6].

[8] Lee Gillam, Neil Newbold, and Neil Cooke. Educated Guesses and Equality Judgements: Using Search Engines and Pairwise Match for External Plagiarism Detection—Notebook for PAN at CLEF 2012. In Forner et al. [5]. ISBN 978-88-904810-3-1. URL http://www.clef-initiative.eu/publication/working-notes.

[9] Tim Gollub, Benno Stein, and Steven Burrows. Ousting Ivory Tower Research: Towards a Web Framework for Providing Experiments as a Service. In Bill Hersh, Jamie Callan, Yoelle Maarek, and Mark Sanderson, editors, *35th International ACM Conference on Research and Development in Information Retrieval (SIGIR 12)*, pages 1125–1126. ACM, August 2012. ISBN 978-1-4503-1472-5. doi: http://dx.doi.org/10.1145/2348283.2348501.

[10] Gaston H. Gonnet, Ricardo A. Baeza-Yates, and Tim Snider. New indices for text: Pat trees and pat arrays. In *Information Retrieval: Data Structures & Algorithms*, pages 66–82. 1992.

[11] Matthias Hagen and Benno Stein. Applying the User-over-Ranking Hypothesis to Query Formulation. In *Advances in Information Retrieval Theory. 3rd International Conference on the Theory of Information Retrieval (ICTIR 11)*, volume 6931 of *Lecture Notes in Computer Science*, pages 225–237, Berlin Heidelberg New York, 2011. Springer. doi: http://dx.doi.org/10.1007/978-3-642-23318-0_21.

[12] Matthias Hagen and Benno Stein. Candidate Document Retrieval for Web-Scale Text Reuse Detection. In *18th International Symposium on String Processing and Information Retrieval (SPIRE 11)*, volume 7024 of *Lecture Notes in Computer Science*, pages 356–367, Berlin Heidelberg New York, 2011. Springer. doi: http://dx.doi.org/10.1007/978-3-642-24583-1_35.

[13] Osama Haggag and Smhaa El-Beltagy. Plagiarism Candidate Retrieval Using Selective Query Formulation and Discriminative Query Scoring—Notebook for PAN at CLEF 2013. In Forner et al. [6].

[14] Marti A. Hearst. TextTiling: Segmenting text into multi-paragraph subtopic passages. *Computational Linguistics*, 23(1):33–64, March 1997. ISSN 0891-2017. URL http://acl.ldc.upenn.edu/J/J97/J97-1003.pdf.

[15] Arun kumar Jayapal. Similarity Overlap Metric and Greedy String Tiling at PAN 2012: Plagiarism Detection—Notebook for PAN at CLEF 2012. In Forner et al. [5]. ISBN 978-88-904810-3-1. URL http://www.clef-initiative.eu/publication/working-notes.

[16] Arun Kumar Jayapal and Binayak Goswami. Submission to the 5th International Competition on Plagiarism Detection. http://www.webis.de/research/events/pan-13, 2013. URL http://www.clef-initiative.eu/publication/working-notes. From Nuance Communications, USA.

[17] Leilei Kong, Haoliang Qi, Shuai Wang, Cuixia Du, Suhong Wang, and Yong Han. Approaches for Candidate Document Retrieval and Detailed Comparison of Plagiarism Detection—Notebook for PAN at CLEF 2012. In Forner et al. [5]. ISBN 978-88-904810-3-1. URL http://www.clef-initiative.eu/publication/working-notes.

[18] Leilei Kong, Haoliang Qi, Cuixia Du, Mingxing Wang, and Zhongyuan Han. Approaches for Source Retrieval and Text Alignment of Plagiarism Detection—Notebook for PAN at CLEF 2013. In Forner et al. [6].

[19] Taemin Lee, Jeongmin Chae, Kinam Park, and Soonyoung Jung. CopyCaptor: Plagiarized Source Retrieval System using Global Word frequency and Local Feedback—Notebook for PAN at CLEF 2013. In Forner et al. [6].

[20] Donald Metzler and W. Bruce Croft. Combining the language model and inference network approaches to retrieval. *Inf. Process. Manage.*, 40(5):735–750, September 2004. ISSN 0306-4573. doi: 10.1016/j.ipm.2004.05.001.

[21] Alireza Nourian. Submission to the 5th International Competition on Plagiarism Detection. http://www.webis.de/research/events/pan-13, 2013. URL http://www.clef-initiative.eu/publication/working-notes. From the Iran University of Science and Technology.

[22] Gabriel Oberreuter, David Carrillo-Cisneros, Isaac D. Scherson, and Juan D. Velásquez. Submission to the 4th International Competition on Plagiarism Detection. http://www.webis.de/research/events/pan-12, 2012. ISSN 2038-4963. URL http://www.clef-initiative.eu/publication/working-notes. From the University of Chile, Chile, and the University of California, USA.

[23] Yurii Palkovskii and Alexei Belov. Applying Specific Clusterization and Fingerprint Density Distribution with Genetic Algorithm Overall Tuning in External Plagiarism Detection—Notebook for PAN at CLEF 2012. In Forner et al. [5]. ISBN 978-88-904810-3-1. URL http://www.clef-initiative.eu/publication/working-notes.

[24] Yurii Palkovskii and Alexei Belov. Using Hybrid Similarity Methods for Plagiarism Detection—Notebook for PAN at CLEF 2013. In Forner et al. [6].

[25] Martin Potthast. *Technologies for Reusing Text from the Web*. Dissertation, Bauhaus-Universität Weimar, December 2011. URL http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:gbv:wim2-20120217-15663.

[26] 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 09 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.

[27] Martin Potthast, Alberto Barrón-Cedeño, Andreas Eiselt, Benno Stein, and Paolo Rosso. Overview of the 2nd International Competition on Plagiarism Detection. In Martin Braschler, Donna Harman, and Emanuele Pianta, editors, *Working Notes Papers of the CLEF 2010 Evaluation Labs*, September 2010. ISBN 978-88-904810-2-4. URL http://www.clef-initiative.eu/publication/working-notes.

[28] Martin Potthast, Benno Stein, Alberto Barrón-Cedeño, and Paolo Rosso. An Evaluation Framework for Plagiarism Detection. In Chu-Ren Huang and Dan Jurafsky, editors, *23rd International Conference on Computational Linguistics (COLING 10)*, pages 997–1005, Stroudsburg, Pennsylvania, August 2010. Association for Computational Linguistics.

[29] Martin Potthast, Andreas Eiselt, Alberto Barrón-Cedeño, Benno Stein, and Paolo Rosso. Overview of the 3rd International Competition on Plagiarism Detection. In Vivien Petras, Pamela Forner, and Paul D. Clough, editors, *Working Notes Papers of the CLEF 2011 Evaluation Labs*, September 2011. ISBN 978-88-904810-1-7. URL http://www.clef-initiative.eu/publication/working-notes.

[30] Martin Potthast, Tim Gollub, Matthias Hagen, Jan Graßegger, Johannes Kiesel, Maximilian Michel, Arnd Oberländer, Martin Tippmann, Alberto Barrón-Cedeño, Parth Gupta, Paolo Rosso, and Benno Stein. Overview of the 4th International Competition on Plagiarism Detection. In Pamela Forner, Jussi Karlgren, and Christa Womser-Hacker, editors, *Working Notes Papers of the CLEF 2012 Evaluation Labs*, September 2012. ISBN 978-88-904810-3-1. URL http://www.clef-initiative.eu/publication/working-notes.

[31] Martin Potthast, Matthias Hagen, Benno Stein, Jan Graßegger, Maximilian Michel, Martin Tippmann, and Clement Welsch. ChatNoir: A Search Engine for the ClueWeb09 Corpus. In Bill Hersh, Jamie Callan, Yoelle Maarek, and Mark Sanderson, editors, *35th International ACM Conference on Research and Development in Information Retrieval (SIGIR 12)*, page 1004. ACM, August 2012. ISBN 978-1-4503-1472-5. doi: http://dx.doi.org/10.1145/2348283.2348429.

[32] Martin Potthast, Matthias Hagen, Michael Völske, and Benno Stein. Crowdsourcing Interaction Logs to Understand Text Reuse from the Web. In *51st Annual Meeting of the Association of Computational Linguistics (ACL 13) (to appear)*. ACM, August 2013. doi: http://dx.doi.org/.

[33] Stephen E. Robertson, Hugo Zaragoza, and Michael J. Taylor. Simple BM25 extension to multiple weighted fields. In *Proceedings of the 2004 ACM CIKM International Conference on Information and Knowledge Management, Washington, DC, USA, November 8-13, 2004*, pages 42–49, 2004.

[34] Diego A. Rodríguez Torrejón and José Manuel Martín Ramos. Text Alignment Module in CoReMo 2.1 Plagiarism Detector—Notebook for PAN at CLEF 2013. In Forner et al. [6].

[35] Mehrin Saremi and Farzin Yaghmaee. Submission to the 5th International Competition on Plagiarism Detection. http://www.webis.de/research/events/pan-13, 2013. URL http://www.clef-initiative.eu/publication/working-notes. From Semnan University, Iran.

[36] Jacob Shapiro and Isak Taksa. Constructing web search queries from the user's information need expressed in a natural language. In *Proceedings of the 2003 ACM Symposium on Applied Computing (SAC), March 9-12, 2003, Melbourne, FL, USA*, pages 1157–1162. ACM, 2003.

[37] Prasha Shrestha and Thamar Solorio. Using a Variety of n-Grams for the Detection of Different Kinds of Plagiarism—Notebook for PAN at CLEF 2013. In Forner et al. [6].

[38] Efstathios Stamatatos. Plagiarism Detection Using Stopword *n*-Grams. *JASIST*, 62(12):2512–2527, 2011. doi: http://dx.doi.org/10.1002/asi.21630.

[39] Benno Stein and Matthias Hagen. Introducing the User-over-Ranking Hypothesis. In *Advances in Information Retrieval. 33rd European Conference on IR Resarch (ECIR 11)*, volume 6611 of *Lecture Notes in Computer Science*, pages 503–509, Berlin Heidelberg New York, April 2011. Springer. doi: http://dx.doi.org/10.1007/978-3-642-20161-5_50.

[40] Benno Stein, Sven Meyer zu Eißen, and Martin Potthast. Strategies for Retrieving Plagiarized Documents. In Charles Clarke, Norbert Fuhr, Noriko Kando, Wessel Kraaij, and Arjen P. de Vries, editors, *30th International ACM Conference on Research and Development in Information Retrieval (SIGIR 07)*, pages 825–826, New York, July 2007. ACM. ISBN 987-1-59593-597-7. doi: http://dx.doi.org/10.1145/1277741.1277928.

[41] Šimon Suchomel, Jan Kasprzak, and Michal Brandejs. Three Way Search Engine Queries with Multi-feature Document Comparison for Plagiarism Detection—Notebook for PAN at CLEF 2012. In Forner et al. [5]. ISBN 978-88-904810-3-1. URL http://www.clef-initiative.eu/publication/working-notes.

[42] Šimon Suchomel, Jan Kasprzak, and Michal Brandejs. Diverse Queries and Feature Type Selection for Plagiarism Discovery—Notebook for PAN at CLEF 2013. In Forner et al. [6].

[43] Ondřej Veselý, Tomáš Foltýnek, and Jiří Rybička. Source Retrieval via Naïve Approach and Passage Selection Heuristics—Notebook for PAN at CLEF 2013. In Forner et al. [6].

[44] Kyle Williams, Hung-Hsuan Chen, Sagnik Ray Chowdhury, and C. Lee Giles. Unsupervised Ranking for Plagiarism Source Retrieval—Notebook for PAN at CLEF 2013. In Forner et al. [6].