Overview of the 6th International Competition on Plagiarism Detection

Martin Potthast,[1] Matthias Hagen,[1] Anna Beyer,[1] Matthias Busse,[1]
Martin Tippmann,[1] Paolo Rosso,[2] and Benno Stein[1]

[1]Web Technology & Information Systems, Bauhaus-Universität Weimar, Germany
[2]Natural Language Engineering Lab, Universitat Politècnica de València, Spain

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

Abstract This paper overviews 17 plagiarism detectors that have been evaluated within the sixth international competition on plagiarism detection at PAN 2014. We report on their performances for the two tasks source retrieval and text alignment of external plagiarism detection. For the third year in a row, we invite software submissions instead of run submissions for this task, which allows for cross-year evaluations. Moreover, we introduce new performance measures for text alignment to shed light on new aspects of detection performance.

1 Introduction

Algorithms for the retrieval and extraction of text reuse from large document collections are central to applications such as plagiarism detection, copyright protection, and information flow analysis. They have to be able to deal with all kinds of text reuse ranging from verbatim copies and quotations to paraphrases and translations to summaries [21]. Particularly the latter kinds of text reuse still present a formidable challenge to both engineering and evaluation of retrieval and extraction algorithms. Until recently, one of the primary obstacles to the development of new algorithms has been a lack of evaluation resources. To rectify this lack, we have build a variety of high-quality, large-scale evaluation resources [29, 27], which have been employed within our annual shared tasks on plagiarism detection since 2009, whereas this paper reports on the results of our shared task's sixth edition.[1]

Since the plagiarism detection task has been running for six years in a row, we observe a multi-year life cycle within this shared task. It can be divided into three phases, namely an innovation phase, a consolidation phase, and a production phase. In the innovation phase, new evaluation resources are being developed and introduced for the first time, such as new corpora, new performance measures, and new technologies. The introduction of new evaluation resources typically stirs up a lot of dust and is prone to errors and inconsistencies that may spoil evaluation results to some extent. This cannot be avoided, since only the use of new evaluation resources by many different parties

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

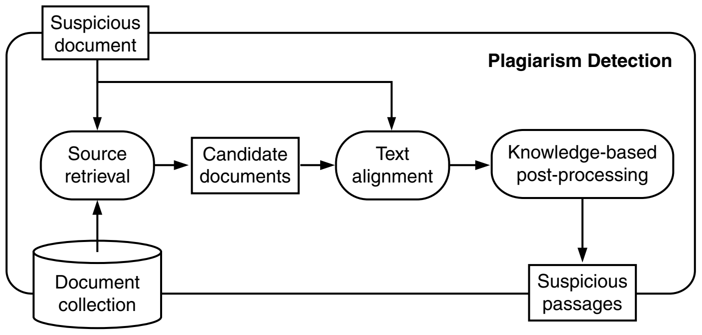

Figure 1. Generic retrieval process to detect plagiarism [37].

will reveal their shortcomings. Therefore, the evaluation resources are released only sparingly so they last for the remainder of a cycle. In the consolidation phase, based on the feedback and results obtained from the first phase, the new evaluation resources are developed to maturity by making adjustments and fixing errors. In the production phase, the task is repeated with little changes to allow participants to build upon and to optimize against what has been accomplished, and, to make the most of the prior investment in developing the new evaluation resources. Meanwhile, new ideas are being developed to introduce further innovation.

1.1 Terminology and Related Work

Terminology. 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 source documents $D_{\text{src}} \subseteq D$ that are likely sources for plagiarism regarding d_{plg}. Second, text alignment, where each candidate source 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 inspection.

Shared Tasks on Plagiarism Detection. We have organized shared tasks on plagiarism detection annually since 2009. In the innovation phase of our shared task at PAN 2009 [30], we developed the first standardized evaluation framework for plagiarism detection [29]. This framework was consolidated in the second and third task at PAN 2010 and 2011 [22, 23], and it has since entered the production phase while being adopted by the community. Our initial goal with this framework was to evaluate

[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.

the process of plagiarism detection depicted in Figure 1 as a whole. We expected that participants would implement source retrieval algorithms as well as text alignment algorithms and use them as modules in their plagiarism detectors. However, the results of the innovation phase proved otherwise, since participants implemented only text alignment algorithms, whereas they resorted to exhaustively comparing all pairs of documents within our evaluation corpora, even when the corpora were tens of thousands of documents large. Therefore, upon entering the production phase after the third edition of our shared task, and, because of continued interest from the research community, we rechristened the shared task to text alignment and continued to offer it in the three following years at PAN 2012 to 2014.

To establish source retrieval as a shared task of its own, we introduced it at PAN 2012 next to the text alignment task [24], thus entering a new task life cycle for this task. We developed a new, large-scale evaluation corpus of essay-length plagiarism cases that have been written manually, and whose sources have been retrieved manually from the ClueWeb corpus [27]. Given our above observation from the text alignment task, the ClueWeb was deemed too large to be exhaustively compared to a given suspicious document in a reasonable time. Furthermore, we developed a new search engine for the ClueWeb called ChatNoir [26], which serves participants who do not wish to develop their own ClueWeb search engine as a means of participation. We then offered source retrieval as an individual task based on the new evaluation resources [24, 25], whereas this year marks the third time we do so, and the transition of the source retrieval task into the production phase.

1.2 Contributions

Both source retrieval and text alignment are now in the production phase of their life cycles. Therefore, we refrain from changing the existing evaluation resources too much, whereas we continue to maintain them. Therefore, our contributions this year consist of (1) a survey of submitted approaches, which reveals new trends among participants at solving the respective tasks, and (2) an analysis of the participants' retrieval performances on a per-obfuscation basis, using new performance measures, and in direct comparison to participants from previous years.

In this connection, our goal with both shared tasks is to further automate them. Hence, we continue to develop the TIRA evaluation platform [10, 11], which gives rise to software submissions with minimal organizational overhead. Last year, we focused on TIRA's infrastructure, which employs virtualization technology to allow participants to use their preferred development environment, and to secure the execution of untrusted software while making the release the test corpora unnecessary [9]. This year, we introduce a fully-fledged web service as a user interface that enables participants to remote control their evaluations on the test corpora under our supervision.

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

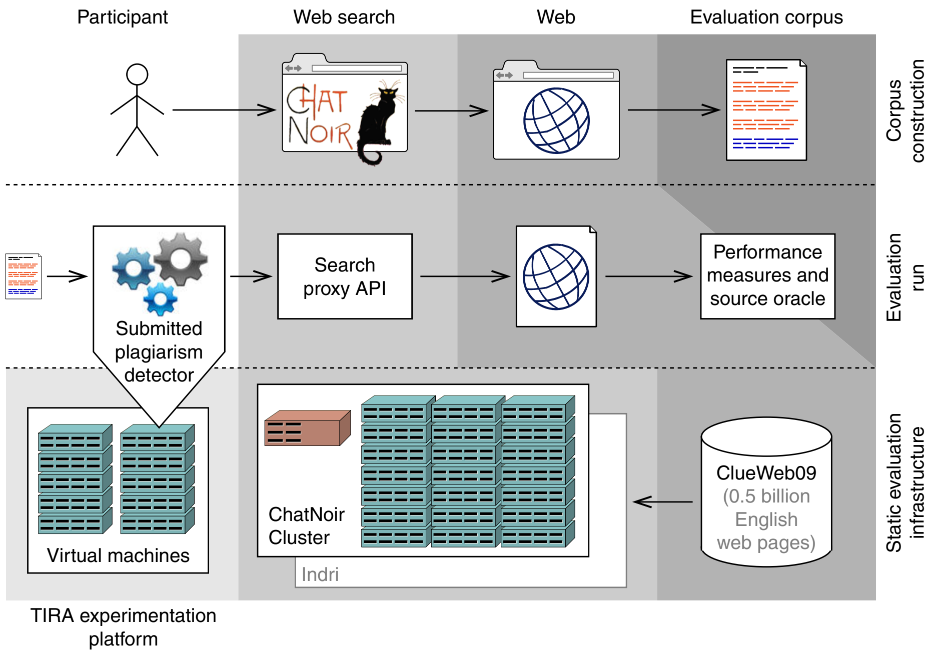
Participant
Web search
Web
Evaluation corpus
Corpus construction
CHAT NOIR
Submitted plagiarism detector
Search proxy API
Performance measures and source oracle
Evaluation run
Virtual machines
ChatNoir Cluster
Indri
ClueWeb09 (0.5 billion English web pages)
Static evaluation infrastructure
TIRA experimentation platform

Figure 2. Overview of the building blocks used in the evaluation of the source retrieval subtask. 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.

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 briefly describe the building blocks of our evaluation setup, provide brief details about the evaluation corpus, and discuss the performance measures (see last year's task overview for more details on these three points [25]). We also survey the submitted softwares, and finally, report on their achieved results in this year's setup.

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. 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. The setup was described in much more detail in last year's task overview [25].

Two main components are the TIRA experimentation platform and the ClueWeb09 with two associated search engines. TIRA [10] itself consists of a number of building blocks; 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 corpus 2009 (ClueWeb09)[3] is one of the most widely adopted web crawls which regularly used for large-scale web search-related evaluations. It consists of about one billion web pages, half of which are English ones. Although an updated version of the corpus has been released,[4] our evaluation is still based on the 2009 version since our corpus of suspicious documents was built on top of ClueWeb09. Indri[5] and ChatNoir [26] are currently the only publicly available search engines that index the ClueWeb09 corpus; their retrieval models are based on language modeling and BM25F, respectively. 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) [28, 27]. 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.

In the last years, we sampled 98 documents from the Webis-TRC-2012 as training and test documents. This year, these documents were provided for training, and another 99 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 terms of precision and recall of the retrieved documents D_{ret} taking into account the effect of near-duplicate web documents as follows (cf. last year's task overview [25] for more details).

For any $d_{\mathrm{ret}} \in D_{\mathrm{ret}}$, we employ a near-duplicate detector to judge whether it is a true positive detection; i.e., whether there is a $d_{\mathrm{src}} \in D_{\mathrm{src}}$ of d_{plg} that is a near-duplicate of d_{ret}. We say that d_{ret} is a true positive detection for a given pair of d_{src} and d_{plg} iff (1) $d_{\mathrm{ret}} = d_{\mathrm{src}}$ (equality), or (2) the Jaccard similarity of the word n-grams in d_{ret} and d_{src} is above 0.8 for $n = 3$, above 0.5 for $n = 5$, and above 0 for $n = 8$ (similarity), or (3) the passages in d_{plg} known to be reused from d_{src} are contained in d_{ret} (containment). Here, containment is measured as asymmetrical set overlap of the

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

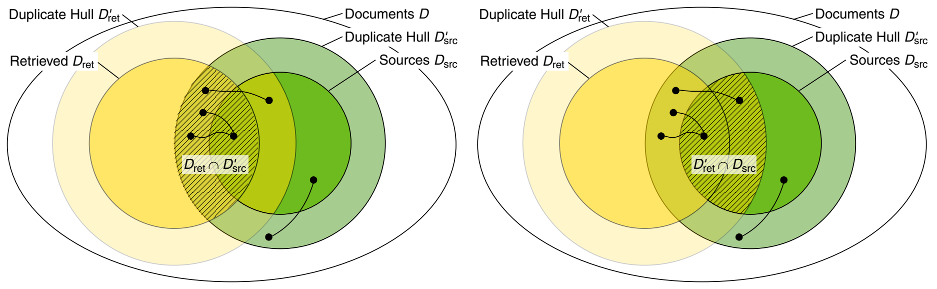
Duplicate Hull D'_{ret}
Retrieved D_{ret}
Documents D
Duplicate Hull D'_{src}
Sources D_{src}
$D_{\text{ret}} \cap D'_{\text{src}}$
Duplicate Hull D'_{ret}
Retrieved D_{ret}
Documents D
Duplicate Hull D'_{src}
Sources D_{src}
$D'_{\text{ret}} \cap D_{\text{src}}$

Figure 3. Effect of near-duplicates on computing precision (left) and recall (right) of retrieved source documents. Without taking near-duplicates into account, a lot of potentially correct sources might be missed.

passages' set of word 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 three-way approach of determining true positive detections inherently entails inaccuracies. While there is no straightforward way to solve this problem, this error source affects all detectors, 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. Note that every d_{src} may have more than one such near-duplicate and every d_{dup} may be a near-duplicate of more than one source document. Further, let D'_{src} 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}:

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

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

$$prec = \frac{|D_{\text{ret}} \cap D'_{\text{src}}|}{|D_{\text{ret}}|}, \qquad rec = \frac{|D'_{\text{ret}} \cap D_{\text{src}}|}{|D_{\text{src}}|}.$$

Rationale for this definition is 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. A further graphical explanation of how we take near-duplicates into account for precision and recall is given in Figure 3. Note that D_{ret} as defined above does not actually contain all duplicates of the retrieved documents, but only those that are already part of D_{src}.

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 To allow for participation in the source retrieval task without the need of having a text alignment component at hand, we provide a source oracle that 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. Hence, applying a custom text alignment strategy can still be beneficial.

2.4 Survey of Retrieval Approaches

Six of the 16 participants submitted softwares for the source retrieval task, all 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 years' source retrieval algorithms: (1) chunking, (2) keyphrase extraction, (3) query formulation, (4) search control, and (5) download filtering. Some participants only slightly changed their approach from the previous year; in what follows, we describe the employed 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) [38], 50-line chunks [5], headings as separate chunks [38], headings to split documents into chunks [38], 100-word chunks based on heading detection [31], 200-word chunks [31], 5-sentence chunks [41, 16], and combinations thereof.

Note that chunks typically are stated as non-overlapping. The potentially interesting question of whether overlapping chunks might help was not really tackled by any approach. However, typical plagiarism cases have no fixed length and overlapping chunks might reduce the risk of, for instance, having more than one source in one chunk of 50 lines. 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. Maybe mixed chunking strategies as seen in Suchomel and Brandejs [38]' approach is an interesting future direction. Notably, their document level queries seem to also guarantee an early recall (cf. Section 2.5).

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.

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. Phrasal search was provided by the Indri search engine. All participants did use Indri when submitting phrasal queries; some of which also combine phrases with non-phrasal ChatNoir queries, the search engine that the original essay authors had used. In particular, Elizalde [5] applies three different keyphrase extraction strategies very similar to her last year's approach: (1) one query per 50-lines chunk containing the top 10 words scored by $tf \cdot idf$ values, (2) first 8-gram with three words from 1 per chunk, (3) 15 phrases based on head noun clusters [3]. Prakash and Saha [31] use the top 5 document-level tf-ranked terms and five paragraph-level tf-ranked terms and the nouns from sentence subgroups to form queries. Kong et al. [16] choose the ten best phrases per chunk according to an own keyphrase extraction based on BM25 and $tf \cdot idf$ weighting. Williams et al. [41] use their very simplistic keyphrase extraction strategy from last year: only nouns, adjectives, and verbs form the keyphrases. Zubarev and Sochenkov [42] follow a similar strategy; for 83 high-weighting sentences (weighting according to overlap with other sentences) they form queries by ignoring articles, pronouns, prepositions, and repeated words. One problem with sentence weighting might be that it does not distribute the selected sentences over the entire document such that for specific parts no keywords might used.

Suchomel and Brandejs [38] apply three different strategies very similar to their last year's approach. First, from the whole document, they use the top 6 words ranked by $tf{\cdot}idf$ values. These top 6 keywords are then also combined with their most frequent two or three term collocations. Second, they use the longest sentence from each paragraph. Third, they detect headers in the text and use 6-term phrases from these headers.

Altogether, the participants' approaches to keyphrase extraction can still 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 $tf \cdot idf$-ranked top scoring words or phrases relying on some background collection for document frequencies. (3) Notably, established keyphrase extraction schemes developed from the respective research community are only used in one approach. (4) Some participants do not rely on one strategy alone but combine the other three 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 Interestingly, most of the participants hardly combine keyphrases into one query apart from merging, for instance, the top k $tf \cdot idf$-ranked terms, then the next k terms, etc. This way, most participants implicitly formulate non-overlapping queries (i.e., they do not explicitly use the same keyword in more than one query) except for some of the participants who basically use all the longer n-grams in the suspicious document or who do not mind same keywords in different queries that appear rather "by accident" than by intention. This non-overlap-approach 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 using overlapping queries [14]. Also note that hardly any 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 sets of keywords or keyphrases 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.

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 participants do not describe a search control. The ones who do basically schedule queries and drop some of the previously generated queries. Prakash and Saha [31] drop a query when more than 60% of its terms are contained in another query. Suchomel and Brandejs [38] 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. Also Zubarev and Sochenkov [42] remove queries for sentences that already are mapped to a potential retrieved source.

Note that still (just as last year) 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. Another interesting aspect might be the scheduling of the queries themselves. The experimental results (cf. Section 2.5) seem to suggest that some document-level queries in the first submission positions guarantee an early recall (e.g., Suchomel and Brandejs [38]). Simply scheduling queries in the order of chunks in the documents instead, might run into problems with early recall as maybe there is not that much reused text at the beginning of a document. This might also be an interesting point for future research.

Download Filtering Given a set of search engine results, 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 [5] focuses on the top 10 results of a query and downloads a result document when at least 90% of the 4-grams in a 500-character snippet are contained in the suspicious document. Prakash and Saha [31] download a document from the top 10 when at least one 5-gram in a 500-character snippet is contained in the suspicious document; they explicitly avoid double downloads of the same URL. Zubarev and Sochenkov [42] base their strategy on the top 7 results and compute similarities of snip-

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

Software Submission Team (alphabetical order)	Year	**Downloaded Sources** F$_1$	Prec.	Rec.	**Total Workload** Queries	Dwlds	**Workload to 1st Detection** Queries	Dwlds	**No Detect.**	**Runtime**
Elizalde	2013	0.16	0.12	0.37	41.6	83.9	18.0	18.2	4	11:18:50
Elizalde	2014	0.34	0.40	0.39	54.5	33.2	16.4	3.9	7	04:02:00
Foltynek	2013	0.11	0.08	0.26	166.8	72.7	180.4	4.3	32	152:26:23
Gillam	2013	0.06	0.04	0.15	**15.7**	86.8	16.1	28.6	34	**02:24:59**
Haggag	2013	0.38	**0.67**	0.31	41.7	**5.2**	13.9	**1.4**	12	46:09:21
Kong	2013	0.01	0.01	**0.59**	47.9	5185.3	**2.5**	210.2	**0**	106:13:46
Kong	2014	0.12	0.08	0.48	83.5	207.1	85.7	24.9	6	24:03:31
Lee	2013	0.40	0.58	0.37	48.4	10.9	6.5	2.0	9	09:17:10
Prakash	2014	0.39	0.38	0.51	60.0	38.8	8.1	3.8	7	19:47:45
Suchomel	2013	0.05	0.04	0.23	17.8	283.0	3.4	64.9	18	75:12:56
Suchomel	2014	0.11	0.08	0.40	19.5	237.3	3.1	38.6	2	45:42:06
Williams	2013	**0.47**	0.60	0.47	117.1	12.4	23.3	2.2	7	76:58:22
Williams	2014	**0.47**	0.57	0.48	117.1	14.4	18.8	2.3	4	39:44:11
Zubarev	2014	0.45	0.54	0.45	37.0	18.6	5.4	2.3	3	40:42:18

pet sentences to the suspicious document's sentences. A download was scheduled when the similarity is high. Suchomel and Brandejs [38] obtain snippets for each individual query term and download documents (no information on the number of results per query is given) when more than 20% of the word 2-grams in the concatenated snippets also appear in the suspicious document. Williams et al. [41] try to train a classifier for download scheduling using a lot of search engine and snippet features. However, comparing their approach from last year with a more simplistic download filtering and this year's classifier idea, not much improvement was achieved (basically the same number of downloads and hardly any improvements in overall or early recall). Kong et al. [16] simply download the top 3 results per query.

Interestingly, the participants heavily rely on the retrieval models of the search engines by focusing on the at most top 10 results per query. It is probably not much more expensive to get more results per query to be able to select from a wider range of results. Interestingly, Elizalde [5] requests 30 results per query but then immediately focuses on the top 10 documents without any further consideration of the lower ranks. Other participants restrict themselves to only a few documents per query while comparing against maybe a hundred results might not be much more costly than selecting from only 3 results. Considering more results per query might be an interesting option for future research based on the User-over-Ranking hypothesis [36, 13].

2.5 Evaluation Results

Table 1 shows the performances of the six plagiarism detectors that took part in this year's source retrieval subtask as well as those of the last year's participants whose approaches were re-evaluated on this year's test corpus using the TIRA experimentation platform. Since there is currently no single 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

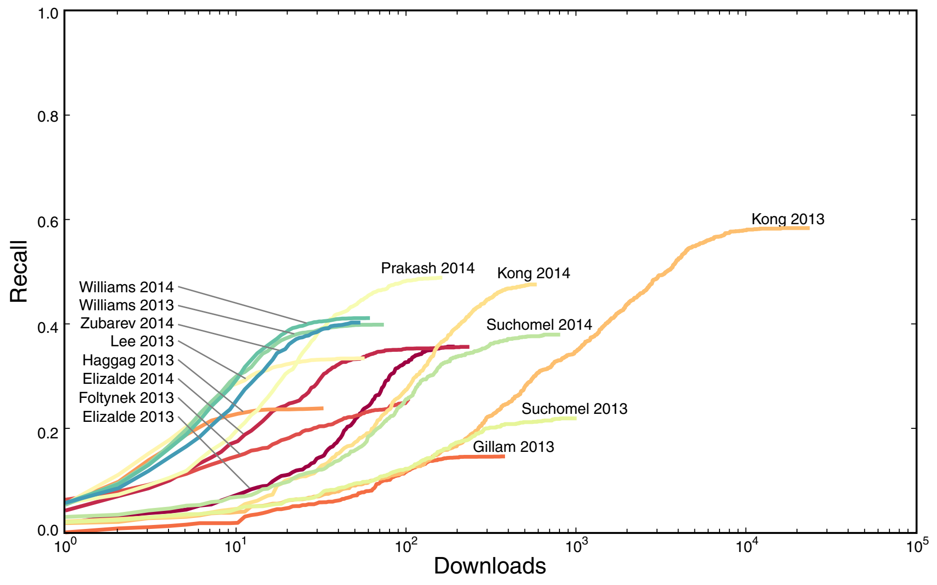

Figure 4. Recall at a specific number of downloads per participant averaged over all topics.

is no single detector that performs best on all accounts. Rather, different detectors have different characteristics.

Arguably, highest possible recall at a reasonable workload (queries and downloads) is the goal of source retrieval. One might discuss for instance, whether recall should be given more weight in the F-measure. Still, when sorting the participants by recall, it appears that only two of the top 8 participants are not from 2014. This indicates some progress in the "right" direction for this task. However, the 2013-approach of Kong et al. [17] still achieves the best recall—at the cost of poor precision. To further shed some light on the recall of the different approaches, Figure 4 shows the recall against the number of downloaded documents. It can be seen that recall is typically gained over the whole process of downloading documents and not with the very first downloads (the plateau effect at the upper right end of each plot is due to the averaging). Unsurprisingly, some of the low-workload approaches achieve higher recall levels with fewer downloads while approaches with more downloads typically achieve their better final recall levels only at a much higher number of downloads. Thus, focusing on download filtering strategies at document-level (when all queries are submitted) might improve their early recall at the download-level—but this would probably harm recall when measured against the submitted queries.

Interestingly, the ensemble of all submitted approaches would achieve an average recall of 0.85 retrieving all sources for 48 topics. Only for 14 topics the recall is below 0.6 (which is the best individual average recall).

Not just focusing on recall, a per-participant analysis also reveals some interesting observations when comparing the approaches from different years. For instance, Suchomel and Brandejs [38] almost doubled their recall without increased query load.

Also Elizalde [5] managed to save a lot of downloads and improve precision over last year without harming recall. By saving a lot of downloads, Kong et al. [16] sacrificed their good recall from last year. A little disappointing is the almost negligible effect of Williams et al. [41] efforts to improve over last year: the better scheduling of the downloads yields no real changes in their overall performance; still, a lot of queries are invested for only a tenth of actual downloads. Prakash and Saha [31] and Zubarev and Sochenkov [42] achieve very good results with their first participation in source retrieval, contributing a lot to the better average recall of this year's approaches over the last year.

The detectors of Williams et al. [40, 41] achieves the best trade-off between precision and recall and therefore the best F_1 value. This detector is followed closely by that of Zubarev and Sochenkov [42], which also achieves almost the same balanced precision and recall with a much smaller workload. It is not easy to decide which of the participating 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 five documents on average per suspicious document and minimizes the time to first detection; however, it also has no detection at all for 12 documents. Despite the excellent precision/recall trade-off of Williams et al.'s detectors, it incurs the second-highest costs in terms of queries on average, much more than some other participants that achieve better or only slightly worse recall. Kong et al.'s detector has the highest download costs, but one may argue that downloads are much cheaper than queries, and that in the source retrieval task recall is more important than precision.

Altogether, the current strategies might be a little too focused on saving downloads (and queries) compared to for instance increased recall. Also runtime should probably not be the key metric to optimize (e.g., using threads instead of sequential processing does not decrease the workload on the search engines). A reasonable assumption probably is that recall is most important to the end user of a source retrieval system. Investing a couple of queries and a couple (maybe even hundreds) of downloads to achieve a recall above 0.8 might be a very important research direction. In the end, whatever source the source retrieval step misses, cannot be found by a later text alignment step. This probably is a key argument for a recall-oriented source retrieval strategy that also takes into account basic considerations on total workload of query submission and downloads. It would be interesting to see efforts in that direction of significantly improved recall at a moderate cost increase in future approaches.

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. To provide a challenging evaluation corpus, we resort to our previous corpus construction efforts. The performance of a plagiarism detector is measured based on the traditionally employed measures plagdet, precision, recall, and granularity, whereas we also introduce new measures that shed light on different performance aspects of plagiarism detection. Finally, we conduct a cross-year evaluation and compare the performance of this year's detectors with those of last year.

3.1 Evaluation Corpus

As an evaluation corpus for this year, we reuse that of last year [25]. The corpus is based on the Webis-TRC-13 [27]. Instead of employing the documents of that corpus directly, pairs of documents that comprise reused passages have been constructed automatically, similarly to previous years [29]. The corpus comprises plagiarism cases whose reused portions of text have been subject to four obfuscation strategies, namely verbatim copies, random obfuscation, cyclic translation obfuscation, and summary obfuscation. The latter strategy has been found to be the most challenging kind of obfuscation in terms of being detected to date.

While the best performing approach was determined based on last year's test corpus, we have compiled a variant of the above corpus using the same methods as before which comprises only unobfuscated and randomly obfuscated plagiarism. This supplemental corpus serves as a baseline corpus.

Discussion Reusing a previously constructed evaluation corpus another time has been a compromise on our part to free up time for working on the front end of the TIRA experimentation platform. This strategy, however, bears the risk of approaches being overfitted to a given corpus and participants cheating. While reusing a previously released corpus would not be a problem if participants used only the training data to fine-tune their approach, it cannot be entirely ruled out that the publicly available test data has been used as well. Therefore, the evaluation results for text alignment of this year must be taken with a grain of salt.

In an attempt to alleviate this problem, we constructed a supplemental corpus using the same corpus construction process that was used for the reused corpus, however, the supplemental corpus comprises only basic obfuscation strategies that can be detected a lot easier than, for example, summary obfuscation. Therefore, this corpus was not chosen to be the reference for this year's ranking among participants.

In general, we are thinking of new ways to organize text alignment as a shared task. Throughout the years, we have created a new evaluation corpus every year, except for this year, adding new kinds of obfuscation strategies each time. However, we feel that this level of output cannot be sustained much longer; should there still be considerable interest in continuing this task, we will involve participants in the corpus construction efforts by submitting a corpus of their own design, while cross-evaluating the submitted corpora using all submitted approaches. We have invited data submissions on a voluntary basis before, however, without much success. Perhaps, by making data submissions a mandatory part of this shared task, there will be more participant engagement.

3.2 Performance Measures Revisited

To assess the performance of the submitted text alignment softwares, we employ the performance measures used in previous evaluations [29]: precision, recall, and granularity, which are combined into the plagdet score. While these measures are not beyond criticism, they have served as a reliable means to rank plagiarism detectors; these measures are generally perceived as very strict.

This year, we revisit performance measurement of plagiarism detectors by shedding light on more abstract levels of detection performance. Until now, plagiarism detection performance has been measured at the character level under the model of a user who expects her plagiarism detector to retrieve and extract contiguous plagiarized passages of text from a given pair of documents. For example, many current plagiarism detectors extract only overlapping substrings of a given plagiarism case which makes reviewing their detections cumbersome, whereas extracting contiguous passages has turned out to be much more convenient. Therefore, the current measures have been specifically developed to capture the "completeness" of detection of a given plagiarism case: they measure the precision and recall of detecting a plagiarism case at character level and combine that with the granularity measure, which counts the number of times a given plagiarism case has been detected.

However, there is more than one relevant user model, and performance can be measured at different levels of abstraction, which may even lead to contrary results as to which particular plagiarism detection approach is best suited for a particular user. In what follows, we review the previous performance measures, and add two new levels of abstraction. They incorporate different assumptions about the detection characteristics preferred by users. Besides the character level, we propose to measure detection performance at the case level, which fixes the minimum precision and recall at which a plagiarism case has to be detected, and, at the document level, which disregards whether all plagiarism cases present in a document are detected as long as a significant portion of one of them is detected. The case level measures assume a user is interested in covering all plagiarism cases which are present in a given collection, whereas the document level measures assume a user wants to determine if a document is suspicious and worth further analysis.

Character level performance measures 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. A plagiarism case $s = \langle s_{\text{plg}}, d_{\text{plg}}, s_{\text{src}}, d_{\text{src}} \rangle$, $s \in S$, 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 $\mathbf{r}$. We say that r detects s iff $\mathbf{s} \cap \mathbf{r} \neq \emptyset$ and s_{plg} overlaps with r_{plg} and s_{src} overlaps with r_{src}. 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., $S_R = \{s \mid s \in S \text{ and } \exists r \in R : r \text{ detects } s\}$ and $R_s = \{r \mid r \in R \text{ and } r \text{ detects } s\}$. 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.

Case level performance measures Let S and R be defined as above. Further, let

$$S' = \{s \mid s \in S \text{ and } rec_{\text{char}}(s, R) > \tau_1 \text{ and } \exists r \in R\text{: } r \text{ detects } s \text{ and } prec_{\text{char}}(S, r) > \tau_2\}$$

denote the subset of all plagiarism cases S which have been detected with more than a threshold τ_1 in terms of character recall rec_{char} and more than a threshold τ_2 in terms of character precision $prec_{\text{char}}$. Likewise, let

$$R' = \{r \mid r \in R \text{ and } prec_{\text{char}}(S, r) > \tau_2 \text{ and } \exists s \in S\text{: } r \text{ detects } s \text{ and } rec_{\text{char}}(s, R) > \tau_1\}$$

denote the subset of all detections R which contribute to detecting plagiarism cases with more than a threshold τ_1 in terms of character recall rec_{char} and more than a threshold τ_2 in terms of character precision $prec_{\text{char}}$. Here, character recall and precision derive from the character level performance measures defined above:

$$prec_{\text{char}}(S, r) = \frac{|\bigcup_{s \in S}(\mathbf{s} \sqcap \mathbf{r})|}{|\mathbf{r}|}, \qquad rec_{\text{char}}(s, R) = \frac{|\bigcup_{r \in R}(\mathbf{s} \sqcap \mathbf{r})|}{|\mathbf{s}|}.$$

Based on this notation, we compute case level precision and recall as follows:

$$prec_{\text{case}}(S, R) = \frac{|R'|}{|R|}, \qquad rec_{\text{case}}(S, R) = \frac{|S'|}{|S|}.$$

The thresholds τ_1 and τ_2 can be used to adjust the minimal detection accuracy with regard to passage boundaries. Threshold τ_1 adjusts how accurate a plagiarism case has to be detected, whereas threshold τ_2 adjusts how accurate a plagiarism detection has to be. Beyond the minimal detection accuracy imposed by these thresholds, however, a higher detection accuracy does not contribute to case level precision and recall. If $\tau_1 \to 1$ and $\tau_2 \to 1$, the minimal required detection accuracy approaches perfection, whereas if $\tau_1 \to 0$ and $\tau_2 \to 0$, it is sufficient to report an entire document as plagiarized to achieve perfect case level precision and recall. In between these extremes, it is an open question which threshold settings are valid with regard to capturing the minimally required detection quality beyond which most users of a plagiarism detection system will not perceive improvements, anymore. Hence, we choose $\tau_1 = \tau_2 = 0.5$ as a reasonable trade off, for the time being: for case level precision, a plagiarism detection r counts a true positive detection if it contributes to detecting at least $\tau_1 = 0.5 \sim 50\%$ of a plagiarism case s, and, if at least $\tau_2 = 0.5 \sim 50\%$ of r contributes to detecting plagiarism cases. Likewise, for case level recall, a plagiarism case s counts as detected if at least 50% of s are detected, and, if a plagiarism detection r contributes to detecting s while at least 50% of r contributes to detecting plagiarism cases in general.

Document level performance measures Let S, R, and R' be defined as above. Further, let D_{plg} be the set of suspicious documents and D_{src} be the set potential source documents. Then $D_{\text{pairs}} = D_{\text{plg}} \times D_{\text{src}}$ denotes the set of possible pairs of documents that a plagiarism detector may analyze, whereas

$$D_{\text{pairs}|S} = \{(d_{\text{plg}}, d_{\text{src}}) \mid (d_{\text{plg}}, d_{\text{src}}) \in D_{\text{pairs}} \text{ and } \exists s \in S : d_{\text{plg}} \in s \text{ and } d_{\text{src}} \in s\}$$

denotes the subset of D_{pairs} whose document pairs contain the plagiarism cases S, and

$$D_{\text{pairs}|R} = \{(d_{\text{plg}}, d_{\text{src}}) \mid (d_{\text{plg}}, d_{\text{src}}) \in D_{\text{pairs}} \text{ and } \exists r \in R : d_{\text{plg}} \in r \text{ and } d_{\text{src}} \in r\}$$

denotes the corresponding subset of D_{pairs} for which plagiarism was detected in R. Likewise, $D_{\text{pairs}|R'}$ denotes the subset of D_{pairs} for which plagiarism was detected when requiring a minimal detection accuracy as per R' defined above. Based on this notation, we compute document level precision and recall as follows:

$$prec_{\text{doc}}(S, R) = \frac{|D_{\text{pairs}|S} \cap D_{\text{pairs}|R'}|}{|D_{\text{pairs}|R}|}, \qquad rec_{\text{doc}}(S, R) = \frac{|D_{\text{pairs}|S} \cap D_{\text{pairs}|R'}|}{|D_{\text{pairs}|S}|}.$$

Again, the thresholds τ_1 and τ_2 allow for adjusting the minimal required detection accuracy for R', but for document level recall, it is sufficient that at least one plagiarism case is detected beyond that accuracy in order for the corresponding document pair $(d_{\text{plg}}, d_{\text{src}})$ to be counted as true positive detection. If none of the plagiarism cases present in $(d_{\text{plg}}, d_{\text{src}})$ is detected beyond the minimal detection accuracy, it is counted as false negative, whereas if detections are made for a pair of documents in which no plagiarism case is present, it is counted as false positive.

Discussion Compared to the character level measures, the case level measures relax the fine-grained measurement of plagiarism detection quality to allow for judging a detection algorithm by its capability of "spotting" plagiarism cases reasonably well with respect to the minimum detection accuracy fixed by the thresholds τ_1 and τ_2. For example, a user who is interested in maximizing case level performance may put emphasis on the coverage of all plagiarism cases rather than the precise extraction of each individual plagiarized pair of passages. The document level measures further relax the requirements to allow for judging a detection algorithm by its capability "to raise a flag" for a given pair of documents, disregarding whether it finds all plagiarism cases contained. For example, a user who is interested in maximizing these measures puts emphasis on being made suspicious which might lead to further, more detailed investigations. In this regard the three levels of performance measurement complement each other. To rank plagiarism detection with regard to their case level performance and their document level performance, we currently use the F_α-Measure. While the best setting of α is also still unclear, we resort to $\alpha = 1$.

3.3 Survey of Text Alignment Approaches

Eleven of the 16 participants submitted softwares that implement text alignment, and for ten of them also a notebook describing their approach has been submitted. An analysis of these notebooks reveals that a number of building blocks 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: most of this year's approaches to text alignment implement the so-called seed and extend-paradigm which is frequently applied in gene sequence alignment. However, also a new trend can be observed, namely methods to predict the obfuscation type at hand and the dynamic choice or adjustment of text alignment approaches based on the prediction. In what follows, we survey the approaches 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 a suspicious and a 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:

– Alvi et al. [2] use character 20-grams and the Rabin-Karp algorithm for string matching between suspicious and source document.
– Glinos [8] use word 1-grams and exact matching. In a supplementary approach, they use only the top 30 most frequent words longer than 5 characters.
– Sanchez-Perez et al. [33] use sentences, whereas short sentences of less than 4 words are joined with their respective succeeding sentence, and they apply a sentence similarity measure where each sentence is represented as $tf \cdot idf$-weighted vector which are compared using cosine similarity and the Dice coefficient. Sentences match if their similarities under both similarity measures exceeds a threshold of 0.33.
– Gross and Modaresi [12] use skip word 2-grams with skips ranging from 1 to 4 and exact matching, whereas seeds appearing more than four times are discarded.
– Rodríguez Torrejón and Martín Ramos [32] reuse their previous approach which is based on sorted 3-grams and sorted 1-skip 3-grams and exact matching.
– Abnar et al. [1] use word 2-grams to word 5-grams. Matching is based on a similarity measure that computes pairwise word similarities between a pair of n-grams, incorporating knowledge about how likely a given word is exchanged by another (e.g., because it is a synonym). This approach differs from others, where, for example, synonyms are normalized and exact matching is applied.
– Palkovskii and Belov [20] use word n-grams, stop word n-grams, named entity n-grams, frequent word n-grams, stemmed n-grams, sorted n-grams, and skip-n-grams and exact matching, however, it is not clear which n is used.
– Gillam and Notley [7] apply a custom fingerprinting approach and compute similarity-sensitive hash values over portions of the input documents, whereas algorithm details and parameter settings remain obscure.
– Kong et al. [16] reuse their previous year's approach using sentences as seeds which match if they exceed a similarity threshold.
– Shrestha et al. [35] also use sentences which are matched if their TER-p score exceeds a threshold, but they also use word 2-grams and exact matching.

Before computing seeds, many participants choose to collapse whitespace, reduce cases, remove non-alphanumeric characters, 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.

The extension algorithms applied this year have become more diverse, including rule-based approaches, dynamic programming, and clustering-based approaches. In previous years, rule-based approaches have been used by almost everyone; they merge 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 rules depend 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. The complexity of the rule sets and their interdependencies has outgrown a casual description, whereas this year's rules are comparably simple. For example, Alvi et al. [2] merge seeds to aligned passages if they are less than 200 chars apart, and Gillam and Notley [7] continue to develop their previous year's approach which used to merge seeds that are less than 900 chars apart. However, given the success of alternative, more dynamic extension algorithms, crafting rule sets by hand appears not to be a competitive approach, anymore.

One of the classical approaches to extension is dynamic programming, and two participants make use of corresponding bioinformatics algorithms: Glinos [8] employ a variant of the Smith-Waterman algorithm, which they tailored to the application for pairs of texts instead of pairs of gene sequences, making improvements in terms of runtime and multiple detections. Oberreuter and Eiselt [19] employ an algorithm from the BLAST family of local gene sequence alignment algorithms. These algorithms can handle noise, which is why Glinos [8] is at liberty to use word 1-grams as seeds, which may help to uncover text similarities that are lost with longer seeds. However, these algorithms may have difficulties to handle heavy re-ordering of phrases and words.

Many participants apply some kind of clustering algorithm or at least algorithms which relate to clustering algorithms. In their secondary approach, Glinos [8] try to identify clusters of frequent topic-related words in order to better pinpoint summaries and highly obfuscated plagiarism, whereas they do not employ one of the standard clustering algorithms but a handcrafted set of rules. Similarly, Sanchez-Perez et al. [33] apply an approach that relates to divisive clustering. They first merge all subsequent seeds using a broad gap threshold into what they call fragments, and then divide the merged fragments until all divided fragments exceed a similarity threshold, in which case the resulting fragments are output as aligned pairs of passaged. By contrast, Gross and Modaresi [12] apply agglomerative single-linkage clustering, where pairs of seeds are merged based on a distance measure, where the merging order follows that of least distance, and the stopping criterion is defined by a distance threshold that may not be exceeded. Abnar et al. [1] apply the density-based clustering algorithm DBSCAN, and Palkovskii and Belov [20] apply what they call "angled ellipse-based graphical clustering," whereas, for the latter, it remains unclear which clustering algorithm is used, exactly, since no reference nor a description is given.

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, whereas a real-world plagiarism detection might attempt to discern reused text that has been properly acknowledged from reused text for which an acknowledgement is missing. Participants, however, use filtering mainly to optimize against the evaluation corpus in order to maximize the performances measured, which is of course impractical, but in the nature of things in a competition.

Alvi et al. [2] discard alignments of less than 200 chars length in the source document and less than 100 chars length in the suspicious document; Glinos [8] discard alignments that contain less than 40 words on both sides; Sanchez-Perez et al. [33] attempt to disambiguate and merge overlapping alignments and discard alignments of less than 150 chars length; Gross and Modaresi [12] discard alignments of less then 15 words length; Abnar et al. [1] and Shrestha et al. [35] discard short alignments based on an unclear threshold; and other participants either do not give details, or they do not filter alignments.

Remarks and New Trends Given the fact that eleven teams participated in text alignment this year, five of whom for the first time, we conclude that there is still a lot of interest in this shared task, and that there is also a lot to be accomplished still. One of the new trends that many participants have picked up simultaneously is that of tailoring their approaches to specific kinds of obfuscation and then to dynamically select the appropriate approach either based on a prediction which kind of obfuscation is at hand in a given pair of to be analyzed documents, or based on an a posteriori decision rule when applying more than one variant at the same time. This development is encouraging as it opens new avenues of research around text alignment, let alone the opportunity to improve significantly over one-fits-all approaches:

– Glinos [8] distinguishes word order-preserving plagiarism from all other kinds and apply their dynamic programming approach and their clustering-based approach at the same time. It is not entirely clear what the decision rule is, or which approach takes precedence over another. The clustering approach, however, turns out to aim at summaries in particular, so that it may be that the dynamic programming algorithm's output takes precedence.
– Sanchez-Perez et al. [33] distinguish summaries from all other kinds and employ two parameter settings for their approach at the same time, one conservative, the other progressive. Afterwards, the decision which of the two outputs obtained is returned is based on the length imbalance between suspicious passage and source passage. If the source passage is more than 3 times longer than the suspicious passage, the progressive settings take precedence.
– Rodríguez Torrejón and Martín Ramos [32] attempted to tune their approach to the evaluation corpus based on hundreds of runs under different parameter settings. As a result, three parameter sets are outlined, one of which has been chosen to compete.

Table 2. Text alignment performances of the 2014 participants on the 2013 test data.

Team	PlagDet	Recall	Precision	Granularity	Runtime
Sanchez-Perez	0.87818	0.87904	0.88168	1.00344	00:25:35
Oberreuter	0.86933	0.85779	0.88595	1.00369	00:05:31
Palkovskii	0.86806	0.82637	0.92227	1.00580	01:10:04
Glinos	0.85930	0.79331	0.96253	1.01695	00:23:13
Shrestha	0.84404	0.83782	0.85906	1.00701	69:51:15
R. Torrejón	0.82952	0.76903	0.90427	1.00278	00:00:42
Gross	0.82642	0.76622	0.93272	1.02514	00:03:00
Kong	0.82161	0.80746	0.84006	1.00309	00:05:26
Abnar	0.67220	0.61163	0.77330	1.02245	01:27:00
Alvi	0.65954	0.55068	0.93375	1.07111	00:04:57
Baseline	0.42191	0.34223	0.92939	1.27473	00:30:30
Gillam	0.28302	0.16840	0.88630	1.00000	00:00:55

- Palkovskii and Belov [20] attempt to predict which of four kinds of obfuscation is at hand, namely no obfuscation, random obfuscation, summaries, and "undefined." Based on the prediction, a choice is made among four corresponding parameter sets. Only little hints are given about the features used for prediction, and it remains entirely unclear how the prediction pipeline works, exactly, what classifiers are used, and how well the prediction performs.
- Kong et al. [16] attempt to predict whether obfuscated or unobfuscated plagiarism is at hand. This distinction corresponds to that of Glinos [8] mentioned above. They employ logistic regression to train a classifier based on lexical similarity features using the training data set of the evaluation corpus. However, it remains unclear whether the entire document pairs are compared using the similarity measures and how the prediction is incorporated into their text alignment approach. Moreover, no analysis of prediction performance is conducted.

While this development is encouraging, it must be noted that the attempts made are still in their infancy and not yet analyzed well enough to form a conclusion whether they yield useful overall performance improvements or not.

3.4 Evaluation Results

In this section, we report on the evaluation of this year's submissions on the aforementioned evaluation corpora. Moreover, we conduct a cross-year evaluation of all softwares submitted since 2012 on the current evaluation corpus. We further differentiate performance with regard to obfuscation strategies to provide insights into how the softwares deal with different strengths of obfuscation. Finally, we compute the performances of all software using the newly proposed performance measures and contrast them with the traditionally applied measures.

Overall Results of 2014 Table 2 shows the overall performances of the eleven plagiarism detectors that implement text alignment and were submitted this year on the 2013 test data. The overall best performing approach is that of Sanchez-Perez et al. [33], followed by that of Oberreuter and Eiselt [19] and Palkovskii and Belov [20]. The former

Table 3. Text alignment performances of the 2014 participants on the supplemental test data.

Team	PlagDet	Recall	Precision	Granularity	Runtime
Palkovskii	0.90779	0.88916	0.92757	1.00027	00:57:15
Oberreuter	0.89268	0.91539	0.87171	1.00051	00:05:37
Sanchez-Perez	0.89197	0.91984	0.86606	1.00026	00:22:10
Glinos	0.88770	0.84511	0.96007	1.01761	00:19:32
Shrestha	0.86806	0.89839	0.84418	1.00381	74:52:47
Gross	0.85500	0.81819	0.92522	1.02187	00:02:49
R. Torrejón	0.84870	0.80267	0.90032	1.00000	00:00:31
Kong	0.83514	0.84156	0.82882	1.00000	00:05:18
Alvi	0.73416	0.67283	0.90081	1.06943	00:04:17
Abnar	0.66377	0.84779	0.54833	1.00455	20:14:51
Baseline	0.64740	0.52838	0.90024	1.04005	00:15:15
Gillam	0.44076	0.29661	0.85744	1.00000	00:00:56

two detectors have balanced precision and recall, while the latter does not. None of the detectors achieve perfect granularity, yet the scores obtained are very reasonable. While the best performing approach this year comes from a first-time participant, the performances of all five newcomers range from very good to poor. One detector's performance does not exceed the baseline. In terms of precision and granularity, the lower-ranked detectors have some shortcomings, whereas performance decreases more or less steadily toward the lower ranks. In terms of runtime, three detectors took more than one hour to finish, one of which took almost three days. The best performing detector of Rodríguez Torrejón and Martín Ramos [32] finishes the 5185 document pairs in less than a minute, whereas the authors claim even faster runtimes on multi-core machines.

Table 3 shows the overall performances of the elven plagiarism detectors on the supplemental test data, which comprises only unobfuscated, and randomly obfuscated plagiarism. These obfuscation strategies have been found to be easier to detect, which explains the generally higher performances. Interestingly, Sanchez-Perez et al. [33] and Palkovskii and Belov [20] switch places, which may be an artifact of the former's focus on detecting summary plagiarism and the latter's focus on detecting verbatim plagiarism. The performance differences, however, are not very big, and the global ranking does not change a lot compared to Table 2. Note that the results of Table 2 determine the best performing approach of 2014, since the 2013 test data pose a much bigger challenge.

Cross-Year Evaluation between 2012 and 2014 Tables 4 to 7 show the performances of all 29 plagiarism detectors submitted since 2012 that implement text alignment on the 2013 test data. The overall performance of the detectors with regard to the plagdet score can be found in Table 4. As can be seen, many of the approaches submitted 2014 significantly improve over the best performing detectors of previous years. Sanchez-Perez et al. [33] takes the lead across all years on the 2013 test data. The best performing detector from previous years from Kong et al. [18] is ranked sixth so that the 2014 participants seem to have raised the bar for future participants significantly. Again, these results must be taken with a grain of salt, since the 2013 test has been available to participants beforehand: although no participants mention in their notebook paper that they also used this corpus to train their approach, it may be the case that part of the

Table 4. Cross-year evaluation of text alignment software submissions from 2012 to 2014 with respect to $plagdet$. The darker a cell, the better the performance compared to the entire column.

Software Submission		**Obfuscation Strategies of the 2013 Evaluation Corpus**				**Entire Corpus**
Team	Year	None	Random	Cyclic translation	Summary	
Sanchez-Perez	2014	0.90032	0.88417	0.88659	0.56070	0.87818
Oberreuter	2014	0.91976	0.86775	0.88118	0.36804	0.86933
Palkovskii	2014	0.96004	0.86495	0.85750	0.27645	0.86806
Glinos	2014	0.96236	0.80623	0.84722	0.62359	0.85930
Shrestha	2014	0.89174	0.86556	0.84384	0.15550	0.84404
Kong	2012	0.87249	0.83242	0.85212	0.43635	0.83679
R. Torrejón	2014	0.93184	0.75378	0.85899	0.35298	0.82952
Oberreuter	2012	0.94170	0.74955	0.84618	0.13208	0.82678
Gross	2014	0.89950	0.80293	0.83825	0.31869	0.82642
R. Torrejón	2013	0.92586	0.74711	0.85113	0.34131	0.82220
Kong	2014	0.83777	0.82300	0.85162	0.43135	0.82161
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
Abnar	2014	0.85124	0.49058	0.67370	0.17148	0.67220
Alvi	2014	0.92693	0.50247	0.54506	0.09032	0.65954
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
Gillam	2014	0.66329	0.05500	0.00403	0.00000	0.28302
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 5. Cross-year evaluation of text alignment software submissions from 2012 to 2014 with respect to precision. The darker a cell, the better the performance compared to the entire column.

Software Submission		**Obfuscation Strategies of the 2013 Evaluation Corpus**				**Entire Corpus**
Team	Year	None	Random	Cyclic translation	Summary	
Glinos	2014	0.96445	0.96951	0.96165	0.96451	0.96253
Nourian	2013	0.92921	0.96274	0.95856	0.99972	0.94707
Jayapal	2012	0.98542	0.95984	0.89590	0.83259	0.94507
Alvi	2014	0.91875	0.94785	0.95984	0.88036	0.93375
Gross	2014	0.91761	0.96000	0.92105	0.94876	0.93272
Baseline		0.88741	0.98101	0.97825	0.91147	0.92939
Palkovskii	2014	0.95584	0.91453	0.89941	0.91315	0.92227
R. Torrejón	2014	0.89901	0.93843	0.90088	0.89793	0.90427
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	2014	0.88097	0.95157	1.00000	0.00000	0.88630
Oberreuter	2014	0.85231	0.90608	0.89977	0.93581	0.88595
Gillam	2012	0.88128	0.95572	0.97273	0.99591	0.88532
Gillam	2013	0.88088	0.95968	0.97273	0.99591	0.88487
Sanchez-Perez	2014	0.83369	0.91015	0.88465	0.99910	0.88168
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
Shrestha	2014	0.82202	0.91098	0.84604	0.93862	0.85906
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	2014	0.78726	0.87003	0.85822	0.96381	0.84006
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
Abnar	2014	0.74910	0.82988	0.76575	0.92946	0.77330
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 6. Cross-year evaluation of text alignment software submissions from 2012 to 2014 with respect to recall. The darker a cell, the better the performance compared to the entire column.

Software Submission		**Obfuscation Strategies of the 2013 Evaluation Corpus**				**Entire Corpus**
Team	Year	None	Random	Cyclic translation	Summary	
Sanchez-Perez	2014	0.97853	0.86067	0.88959	0.41274	0.87904
Oberreuter	2014	0.99881	0.83254	0.86335	0.24455	0.85779
Shrestha	2014	0.97438	0.83161	0.85318	0.08875	0.83782
Palkovskii	2014	0.96428	0.82244	0.82031	0.17672	0.82637
Kong	2012	0.94836	0.77903	0.85003	0.29892	0.82449
Kong	2013	0.90682	0.78682	0.84626	0.30017	0.81344
Kong	2014	0.89521	0.78079	0.84512	0.29636	0.80746
Glinos	2014	0.96028	0.72478	0.76248	0.48605	0.79331
Saremi	2013	0.95416	0.68877	0.80473	0.10209	0.77123
R. Torrejón	2014	0.96715	0.62985	0.82082	0.23149	0.76903
Oberreuter	2012	0.99932	0.65322	0.79587	0.07076	0.76864
Gross	2014	0.90724	0.71884	0.78410	0.20577	0.76622
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
Abnar	2014	0.99110	0.35360	0.60498	0.12200	0.61163
Sánchez-Vega	2012	0.74452	0.43502	0.58133	0.22161	0.56225
Alvi	2014	0.98701	0.36603	0.41988	0.05685	0.55068
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
Gillam	2014	0.53187	0.02832	0.00202	0.00000	0.16840

Table 7. Cross-year evaluation of text alignment software submissions from 2012 to 2014 with respect to granularity. The darker a cell, the better the performance compared to the entire column.

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
Gillam	2014	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
R. Torrejón	2014	1.00000	1.00000	1.00000	1.06024	1.00278
Kong	2012	1.00000	1.00000	1.00000	1.06452	1.00282
Kong	2014	1.00000	1.00000	1.00000	1.07190	1.00309
Kong	2013	1.00000	1.00000	1.00000	1.07742	1.00336
Sanchez-Perez	2014	1.00000	1.00086	1.00081	1.05882	1.00344
Oberreuter	2014	1.00000	1.00000	1.00000	1.07568	1.00369
Palkovskii	2014	1.00000	1.00176	1.00088	1.10112	1.00580
Shrestha	2014	1.00000	1.00630	1.00948	1.06034	1.00701
Glinos	2014	1.00000	1.04037	1.00547	1.05128	1.01695
Sánchez-Vega	2012	1.00394	1.02200	1.03533	1.04523	1.02196
Abnar	2014	1.00332	1.01509	1.00462	1.39130	1.02245
Gross	2014	1.01998	1.03336	1.01466	1.08667	1.02514
Kueppers	2012	1.02687	1.01847	1.01794	1.31061	1.03497
Nourian	2013	1.00092	1.11558	1.00485	1.34234	1.04343
Alvi	2014	1.03731	1.07203	1.10207	1.26966	1.07111
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

good performance of the 2014 participants can be attributed to the fact that they had a priori access to the test data.

Regarding the obfuscation strategies, the detectors' performances correlate with their overall performance. On unobfuscated plagiarism (column "None" in the tables), Palkovskii and Belov [20] and Glinos [8] perform best, beating the performance of the first-ranked Sanchez-Perez et al. [33] by far. On random obfuscation and cyclic translation, however, the latter maintains his lead. On summary obfuscation, the approach of Glinos [8] and that of Suchomel et al. [39] perform best, again, outperforming Sanchez-Perez et al. [33] by far. This hints that the clustering approach of Glinos [8] works rather well, whereas the combination with the Smith-Waterman algorithm provides for a competitive trade off.

Table 5 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 exception to the rule is the detector of Glinos [8] which, for the first time, achieves both best overall precision and a competitive overall ranking with regard to plagdet performance.

Table 6 shows the detectors' performances with regard to recall. Four 2014 participants now outperform the formerly best performing pair of detectors submitted by Kong et al. [18, 17]. Some participants achieve 0.99 recall on unobfuscated plagiarism, whereas the recall on such plagiarism is generally high, even among the low-ranked detectors. Recall performances on random obfuscation and cyclic obfuscation correlate with those on the entire corpus. The best performing detector on summary obfuscation is still that of Suchomel et al. [39], whereas even Glinos [8] performs significantly worse in terms of recall.

Table 7 shows the detectors' performances with regard to granularity. The top fifth of the table entries have unanimously perfect granularity, so that these approaches are ranked alphabetically. Despite their perfect granularity scores, these detectors do not perform well with regard to other measures which may hint that these detectors emphasize granularity performance too much at the expense of recall, precision, and therefore plagdet. With the exception of summary obfuscation and therefore the performance on the entire corpus, the top half of the table shows near-perfect scores. Only summary obfuscation still poses a slight challenge, whereas it appears that granularity is still mostly under control. We repeat our concern, however, that participants often resort to post-retrieval filtering in order to optimize granularity only for the sake of achieving a good ranking, while some admit that they would not do this in practice.

New Performance Measures Table 8 contrasts the character level performance measures, which are traditionally applied to measure the performance of a plagiarism detector, to the new measures introduced above, which measure performance at case level and at document level. The table shows the performances of all detectors that have been evaluated for text alignment since 2012. When comparing the plagdet performances with the F_1 performances of the case level measures and the document level measures,

Table 8. Cross-year evaluation of text alignment software submissions from 2012 to 2014 with respect to performance measures at character level, case level, and document level.

Software Submission		**Character Level**				**Case Level**			**Document Level**		
Team	Year	*plagdet*	*prec*	*rec*	*gran*	*prec*	*rec*	F_1	*prec*	*rec*	F_1
Sanchez-Perez	2014	0.88	0.88	0.88	1.00	0.90	0.91	0.90	0.92	0.91	0.91
Oberreuter	2014	0.87	0.89	0.86	1.00	0.84	0.89	0.87	0.89	0.89	0.89
Palkovskii	2014	0.87	0.92	0.83	1.01	0.90	0.85	0.87	0.90	0.84	0.87
Glinos	2014	0.86	0.96	0.79	1.02	0.90	0.83	0.87	0.93	0.88	0.91
Kong	2012	0.84	0.85	0.82	1.00	0.86	0.85	0.85	0.89	0.85	0.87
Shrestha	2014	0.84	0.86	0.84	1.01	0.91	0.85	0.88	0.94	0.85	0.89
Gross	2014	0.83	0.93	0.77	1.03	0.90	0.86	0.88	0.93	0.85	0.89
Oberreuter	2012	0.83	0.89	0.77	1.00	0.81	0.79	0.80	0.83	0.80	0.81
R. Torrejón	2014	0.83	0.90	0.77	1.00	0.84	0.83	0.83	0.89	0.84	0.86
R. Torrejón	2013	0.83	0.90	0.77	1.00	0.83	0.83	0.83	0.87	0.84	0.85
Kong	2013	0.82	0.83	0.81	1.00	0.85	0.86	0.85	0.89	0.86	0.87
Kong	2014	0.82	0.84	0.81	1.00	0.86	0.85	0.85	0.89	0.85	0.87
Palkovskii	2012	0.79	0.82	0.76	1.00	0.80	0.80	0.80	0.82	0.80	0.81
R. Torrejón	2012	0.79	0.83	0.75	1.00	0.65	0.79	0.72	0.65	0.78	0.71
Suchomel	2013	0.74	0.73	0.77	1.00	0.66	0.83	0.73	0.67	0.82	0.74
Suchomel	2012	0.73	0.84	0.65	1.00	0.76	0.70	0.73	0.77	0.69	0.73
Saremi	2013	0.70	0.87	0.77	1.24	0.59	0.80	0.68	0.82	0.82	0.82
Shrestha	2013	0.70	0.87	0.74	1.22	0.57	0.76	0.65	0.77	0.78	0.77
Abnar	2014	0.67	0.77	0.61	1.02	0.76	0.63	0.69	0.88	0.65	0.75
Alvi	2014	0.66	0.93	0.55	1.07	0.77	0.59	0.67	0.88	0.63	0.73
Kueppers	2012	0.63	0.87	0.51	1.03	0.76	0.59	0.67	0.83	0.64	0.72
Palkovskii	2013	0.62	0.82	0.54	1.07	0.62	0.56	0.59	0.76	0.59	0.66
Nourian	2013	0.58	0.95	0.43	1.04	0.84	0.44	0.58	0.88	0.45	0.59
Sánchez-Vega	2012	0.46	0.40	0.56	1.02	0.35	0.63	0.45	0.64	0.67	0.66
Baseline		0.42	0.93	0.34	1.27	0.42	0.31	0.36	0.55	0.32	0.41
Gillam	2012	0.41	0.89	0.27	1.00	0.92	0.28	0.43	0.93	0.30	0.46
Gillam	2013	0.40	0.88	0.26	1.00	0.92	0.27	0.42	0.93	0.29	0.44
Gillam	2014	0.28	0.89	0.17	1.00	0.91	0.18	0.31	0.92	0.19	0.31
Jayapal	2013	0.27	0.88	0.38	2.91	0.04	0.33	0.07	0.14	0.34	0.20
Jayapal	2012	0.20	0.95	0.22	2.45	0.01	0.17	0.01	0.02	0.19	0.03

they are highly correlated. This is in the nature of things, since it is unlikely that a plagiarism detector that performs poor at character level performs excellent at case level or at document level. However, the rankings still differ. For example, at case level, the detectors of Shrestha et al. [35] and Gross and Modaresi [12] are ranked second and third to that of Sanchez-Perez et al. [33], whereas the detector of Oberreuter and Eiselt [19] looses some ranks. Most of the other detectors maintain their rank relative to the other detectors. It can be followed that the detectors whose ranks are better than before do a sensible job of "spotting" at least 50% of each plagiarism case, whereas, at character level, they are outperformed by other detectors.

At document level, the detector of Glinos [8] catches up with that of Sanchez-Perez et al. [33], whereas those of Shrestha et al. [35] and Gross and Modaresi [12] also gain a few ranks. Interestingly, also detectors that are ranked lower at character level, such as those of Kong et al. [17, 16] and Saremi and Yaghmaee [34] perform significantly better, which hints that these detectors, along with the other top-ranked ones, are useful for raising suspicions about a given pair of documents.

Nevertheless, the detector of Sanchez-Perez et al. [33] dominates all other detectors at all three levels.

Assessing a new performance measures is a difficult task, since at the beginning it remains unclear whether the intuitions that guided their definition are captured well, and whether they actually reveal performance aspects which other measures do not capture well enough. In our case, all of the current plagiarism detectors have been optimized against the character level performance measures, so that it cannot, yet, be told whether it is possible to build a plagiarism detection which outperforms all others, say, at document level, but not so at the other levels. Only time will tell. Moreover, the hyper-parameters τ_1 and τ_2 as well as the weight α used in the F_α-Measure are not yet fixed and subject to ongoing research. Therefore, it would be premature to give the new sets of performance measures precedence over the existing performance measures, which have been already adopted by the community.

4 Conclusion and Outlook

Altogether, the sixth international competition on plagiarism detection at PAN 2014 has been a success: despite being organized for the sixth time in a row, we see steady interest from from the community to further study this task, as is evidenced by the fact that many participants have returned to make a new submissions. Moreover, the two tasks source retrieval and text alignment are picked up by new participants each year, which makes for a steady stream of new input and inspiration at solving these tasks. In total, 16 teams submitted plagiarism detectors, 6 for source retrieval and 11 for text alignment.

Since both source retrieval and text alignment are in the production phase of their shared task life cycles—they are well-defined and all evaluation resources are set up and provide for a challenging testbed—we have refrained from introducing too many changes to the two tasks. Nevertheless, we continuously work to make the maintenance of the evaluation resources for both tasks easier. This pertains particularly to the resources required for source retrieval which include a fully-fledged search engine for ClueWeb corpus.

Moreover, we continue to pursue our goal of automating evaluations within shared tasks by developing the TIRA experimentation platform [11]. TIRA facilitates software submissions, where participants submit their plagiarism detection software to be evaluated at our site [9]. As of this year, the newly introduced web front end for TIRA allows participants to conduct self-service evaluations on the test data of both our shared tasks under our supervision and guidance, whereas the test data remains hidden from direct access from participants.[6] This has allowed us to put participants back in charge of executing their software while the software itself remains in a running state within virtual machines managed by TIRA. Based on this technology, we conduct cross-year evaluations of all plagiarism detectors that have been submitted to our tasks since 2012.

This year, we place emphasis on analyzing the detection performances of the plagiarism detectors by developing new means of visualizing their performance as well as new performance measures that shed light on different performance aspects of plagiarism detection than the traditionally applied measures. This is ongoing research, and our goal is to provide a more in-depth analysis of each plagiarism detector that enters our evaluations.

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. Abnar, S., Dehghani, M., Zamani, H., Shakery, A.: Expanded N-Grams for Semantic Text Alignment—Notebook for PAN at CLEF 2014. In: [4]
2. Alvi, F., Stevenson, M., Clough, P.: Hashing and Merging Heuristics for Text Reuse Detection—Notebook for PAN at CLEF 2014. In: [4]
3. Barker, K., Cornacchia, N.: Using Noun Phrase Heads to Extract Document Keyphrases. In: Hamilton, H.J. (ed.) 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. Lecture Notes in Computer Science, vol. 1822, pp. 40–52. Springer (2000)
4. Cappellato, L., Ferro, N., Halvey, M., Kraaij, W. (eds.): CLEF 2014 Evaluation Labs and Workshop – Working Notes Papers, 15-18 September, Sheffield, UK. CEUR Workshop Proceedings, CEUR-WS.org (2014), http://www.clef-initiative.eu/publication/working-notes
5. Elizalde, V.: Using Noun Phrases and tf-idf for Plagiarized Document Retrieval—Notebook for PAN at CLEF 2014. In: [4]
6. Forner, P., Navigli, R., Tufis, D. (eds.): CLEF 2013 Evaluation Labs and Workshop – Working Notes Papers, 23-26 September, Valencia, Spain (2013), http://www.clef-initiative.eu/publication/working-notes

[6] www.tira.io

7. Gillam, L., Notley, S.: Evaluating Robustness for 'IPCRESS': Surrey's Text Alignment for Plagiarism Detection—Notebook for PAN at CLEF 2014. In: [4]
8. Glinos, D.: A Hybrid Architecture for Plagiarism Detection—Notebook for PAN at CLEF 2014. In: [4]
9. Gollub, T., Potthast, M., Beyer, A., Busse, M., Rangel, F., Rosso, P., Stamatatos, E., Stein, B.: Recent Trends in Digital Text Forensics and its Evaluation. In: Forner, P., Müller, H., Paredes, R., Rosso, P., Stein, B. (eds.) Information Access Evaluation meets Multilinguality, Multimodality, and Visualization. 4th International Conference of the CLEF Initiative (CLEF 13). pp. 282–302. Springer, Berlin Heidelberg New York (Sep 2013)
10. Gollub, T., Stein, B., Burrows, S.: Ousting Ivory Tower Research: Towards a Web Framework for Providing Experiments as a Service. In: Hersh, B., Callan, J., Maarek, Y., Sanderson, M. (eds.) 35th International ACM Conference on Research and Development in Information Retrieval (SIGIR 12). pp. 1125–1126. ACM (Aug 2012)
11. Gollub, T., Stein, B., Burrows, S., Hoppe, D.: TIRA: Configuring, Executing, and Disseminating Information Retrieval Experiments. In: Tjoa, A.M., Liddle, S., Schewe, K.D., Zhou, X. (eds.) 9th International Workshop on Text-based Information Retrieval (TIR 12) at DEXA. pp. 151–155. IEEE, Los Alamitos, California (Sep 2012)
12. Gross, P., Modaresi, P.: Plagiarism Alignment Detection by Merging Context Seeds—Notebook for PAN at CLEF 2014. In: [4]
13. Hagen, M., Stein, B.: 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). Lecture Notes in Computer Science, vol. 6931, pp. 225–237. Springer, Berlin Heidelberg New York (2011)
14. Hagen, M., Stein, B.: Candidate Document Retrieval for Web-Scale Text Reuse Detection. In: 18th International Symposium on String Processing and Information Retrieval (SPIRE 11). Lecture Notes in Computer Science, vol. 7024, pp. 356–367. Springer, Berlin Heidelberg New York (2011)
15. Haggag, O., El-Beltagy, S.: Plagiarism Candidate Retrieval Using Selective Query Formulation and Discriminative Query Scoring—Notebook for PAN at CLEF 2013. In: [6]
16. Kong, L., Han, Y., Han, Z., Yu, H., Wang, Q., Zhang, T., Qi, H.: Source Retrieval Based on Learning to Rank and Text Alignment Based on Plagiarism Type Recognition for Plagiarism Detection—Notebook for PAN at CLEF 2014. In: [4]
17. Kong, L., Qi, H., Du, C., Wang, M., Han, Z.: Approaches for Source Retrieval and Text Alignment of Plagiarism Detection—Notebook for PAN at CLEF 2013. In: [6]
18. Kong, L., Qi, H., Wang, S., Du, C., Wang, S., Han, Y.: Approaches for Candidate Document Retrieval and Detailed Comparison of Plagiarism Detection—Notebook for PAN at CLEF 2012. In: Forner, P., Karlgren, J., Womser-Hacker, C. (eds.) CLEF 2012 Evaluation Labs and Workshop – Working Notes Papers, 17-20 September, Rome, Italy (Sep 2012), http://www.clef-initiative.eu/publication/working-notes
19. Oberreuter, G., Eiselt, A.: Submission to the 6th International Competition on Plagiarism Detection. http://www.webis.de/research/events/pan-14 (2014), http://www.clef-initiative.eu/publication/working-notes, From Innovand.io, Chile
20. Palkovskii, Y., Belov, A.: Developing High-Resolution Universal Multi-Type N-Gram Plagiarism Detector—Notebook for PAN at CLEF 2014. In: [4]
21. Potthast, M.: Technologies for Reusing Text from the Web. Dissertation, Bauhaus-Universität Weimar (Dec 2011), http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:gbv:wim2-20120217-15663
22. Potthast, M., Barrón-Cedeño, A., Eiselt, A., Stein, B., Rosso, P.: Overview of the 2nd International Competition on Plagiarism Detection. In: Braschler, M., Harman, D., Pianta, E. (eds.) Working Notes Papers of the CLEF 2010 Evaluation Labs (Sep 2010), http://www.clef-initiative.eu/publication/working-notes

23. Potthast, M., Eiselt, A., Barrón-Cedeño, A., Stein, B., Rosso, P.: Overview of the 3rd International Competition on Plagiarism Detection. In: Petras, V., Forner, P., Clough, P. (eds.) Working Notes Papers of the CLEF 2011 Evaluation Labs (Sep 2011), http://www.clef-initiative.eu/publication/working-notes
24. Potthast, M., Gollub, T., Hagen, M., Graßegger, J., Kiesel, J., Michel, M., Oberländer, A., Tippmann, M., Barrón-Cedeño, A., Gupta, P., Rosso, P., Stein, B.: Overview of the 4th International Competition on Plagiarism Detection. In: Forner, P., Karlgren, J., Womser-Hacker, C. (eds.) Working Notes Papers of the CLEF 2012 Evaluation Labs (Sep 2012), http://www.clef-initiative.eu/publication/working-notes
25. Potthast, M., Gollub, T., Hagen, M., Tippmann, M., Kiesel, J., Rosso, P., Stamatatos, E., Stein, B.: Overview of the 5th International Competition on Plagiarism Detection. In: Forner, P., Navigli, R., Tufis, D. (eds.) Working Notes Papers of the CLEF 2013 Evaluation Labs (Sep 2013), http://www.clef-initiative.eu/publication/working-notes
26. Potthast, M., Hagen, M., Stein, B., Graßegger, J., Michel, M., Tippmann, M., Welsch, C.: ChatNoir: A Search Engine for the ClueWeb09 Corpus. In: Hersh, B., Callan, J., Maarek, Y., Sanderson, M. (eds.) 35th International ACM Conference on Research and Development in Information Retrieval (SIGIR 12). p. 1004. ACM (Aug 2012)
27. Potthast, M., Hagen, M., Völske, M., Stein, B.: Crowdsourcing Interaction Logs to Understand Text Reuse from the Web. In: Fung, P., Poesio, M. (eds.) Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL 13). pp. 1212–1221. ACL (Aug 2013), http://www.aclweb.org/anthology/P13-1119
28. Potthast, M., Hagen, M., Völske, M., Stein, B.: Exploratory Search Missions for TREC Topics. In: Wilson, M.L., Russell-Rose, T., Larsen, B., Hansen, P., Norling, K. (eds.) 3rd European Workshop on Human-Computer Interaction and Information Retrieval (EuroHCIR 2013). pp. 11–14. CEUR-WS.org (Aug 2013), http://www.cs.nott.ac.uk/ mlw/euroHCIR2013/proceedings/paper3.pdf
29. Potthast, M., Stein, B., Barrón-Cedeño, A., Rosso, P.: An Evaluation Framework for Plagiarism Detection. In: Huang, C.R., Jurafsky, D. (eds.) 23rd International Conference on Computational Linguistics (COLING 10). pp. 997–1005. Association for Computational Linguistics, Stroudsburg, Pennsylvania (Aug 2010)
30. Potthast, M., Stein, B., Eiselt, A., Barrón-Cedeño, A., Rosso, P.: Overview of the 1st International Competition on Plagiarism Detection. In: Stein, B., Rosso, P., Stamatatos, E., Koppel, M., Agirre, E. (eds.) SEPLN 09 Workshop on Uncovering Plagiarism, Authorship, and Social Software Misuse (PAN 09). pp. 1–9. CEUR-WS.org (Sep 2009), http://ceur-ws.org/Vol-502
31. Prakash, A., Saha, S.: Experiments on Document Chunking and Query Formation for Plagiarism Source Retrieval—Notebook for PAN at CLEF 2014. In: [4]
32. Rodríguez Torrejón, D., Martín Ramos, J.: CoReMo 2.3 Plagiarism Detector Text Alignment Module—Notebook for PAN at CLEF 2014. In: [4]
33. Sanchez-Perez, M., Sidorov, G., Gelbukh, A.: A Winning Approach to Text Alignment for Text Reuse Detection at PAN 2014—Notebook for PAN at CLEF 2014. In: [4]
34. Saremi, M., Yaghmaee, F.: Submission to the 5th International Competition on Plagiarism Detection. http://www.webis.de/research/events/pan-13 (2013), http://www.clef-initiative.eu/publication/working-notes, From Semnan University, Iran
35. Shrestha, P., Maharjan, S., Solorio, T.: Machine Translation Evaluation Metric for Text Alignment—Notebook for PAN at CLEF 2014. In: [4]
36. Stein, B., Hagen, M.: Introducing the User-over-Ranking Hypothesis. In: Advances in Information Retrieval. 33rd European Conference on IR Resarch (ECIR 11). Lecture Notes in Computer Science, vol. 6611, pp. 503–509. Springer, Berlin Heidelberg New York (Apr 2011)

37. Stein, B., Meyer zu Eißen, S., Potthast, M.: Strategies for Retrieving Plagiarized Documents. In: Clarke, C., Fuhr, N., Kando, N., Kraaij, W., de Vries, A. (eds.) 30th International ACM Conference on Research and Development in Information Retrieval (SIGIR 07). pp. 825–826. ACM, New York (Jul 2007)
38. Suchomel, Šimon., Brandejs, M.: Heterogeneous Queries for Synoptic and Phrasal Search—Notebook for PAN at CLEF 2014. In: [4]
39. Suchomel, Šimon., Kasprzak, J., Brandejs, M.: Diverse Queries and Feature Type Selection for Plagiarism Discovery—Notebook for PAN at CLEF 2013. In: [6]
40. Williams, K., Chen, H.H., Chowdhury, S., Giles, C.: Unsupervised Ranking for Plagiarism Source Retrieval—Notebook for PAN at CLEF 2013. In: [6]
41. Williams, K., Chen, H.H., Giles, C.: Supervised Ranking for Plagiarism Source Retrieval—Notebook for PAN at CLEF 2014. In: [4]
42. Zubarev, D., Sochenkov, I.: Using Sentence Similarity Measure for Plagiarism Source Retrieval—Notebook for PAN at CLEF 2014. In: [4]