Overview of the 4th International Competition on Plagiarism Detection

Martin Potthast,[1] Tim Gollub,[1] Matthias Hagen,[1] Jan Graßegger,[1] Johannes Kiesel,[1] Maximilian Michel,[1] Arnd Oberländer,[1] Martin Tippmann,[1] Alberto Barrón-Cedeño,[2] Parth Gupta,[3] Paolo Rosso,[3] and Benno Stein[1]

[1]Web Technology & Information Systems
Bauhaus-Universität Weimar, Germany

[2]Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya, Spain

[3]Natural Language Engineering Lab, ELiRF
Universitat Politècnica de València, Spain

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

Abstract This paper overviews 15 plagiarism detectors that have been evaluated within the fourth international competition on plagiarism detection at PAN'12. We report on their performances for two sub-tasks of external plagiarism detection: candidate document retrieval and detailed document comparison. Furthermore, we introduce the PAN plagiarism corpus 2012, the TIRA experimentation platform, and the ChatNoir search engine for the ClueWeb. They add scale and realism to the evaluation as well as new means of measuring performance.

1 Introduction

To plagiarize means to reuse someone else's work while pretending it to be one's own. Text plagiarism is perhaps one of the oldest forms of plagiarism, which, to this day, remains difficult to be identified in practice. Therefore, a lot of research has been conducted to detect plagiarism automatically. However, much of this research lacks proper evaluation, rendering it irreproducible at times and mostly incomparable across papers [24]. In order to alleviate these issues, we have been organizing annual competitions on plagiarism detection since 2009 [22, 23, 25]. For the purpose of these competitions, we developed the first standardized evaluation framework for plagiarism detection, which has been deployed and revised in the past three competitions, in which a total of 32 teams of researchers took part, 9 of whom more than once.

Ideally, an evaluation framework accurately emulates the real world around a given computational task in a controlled laboratory environment. But actually, every evaluation framework models the real world to some extent only, while resting upon certain design choices which affect the generalizability of evaluation results to practice. This is also true for our evaluation framework, which has a number of shortcomings rendering it less realistic, sometimes leading its users to impractical algorithm design. In this year's fourth edition of the plagiarism detection competition, we venture off the beaten track in order to further push the limits of evaluating plagiarism detectors. Our goal is to create a more realistic evaluation framework in the course of the coming years. In this paper, we describe our new framework and overview the 15 approaches submitted to this year's competition.

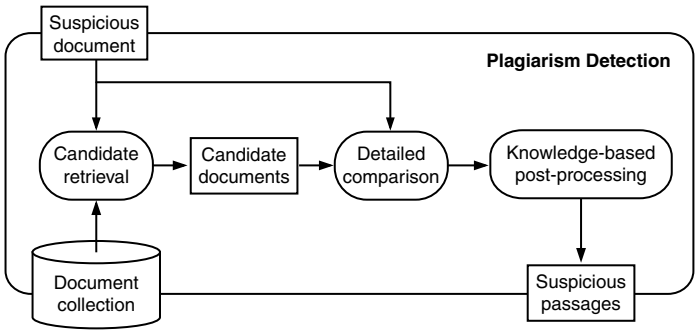

Figure 1. Generic retrieval process to detect plagiarism [31].

1.1 Plagiarism Detection and its Step-wise Evaluation

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

In the previous plagiarism detection competitions, we evaluated external plagiarism detection as a whole, handing out large corpora of suspicious documents and source documents. But instead of following the outlined steps, many resorted to comparing all suspicious documents *exhaustively* to the available source documents. The reason for this was that the number of source documents in our corpus was still too small to justify serious attempts at candidate retrieval. In a realistic setting, however, the source collection is no less than the entire web, which renders exhaustive comparisons infeasible. We hence decided to depart from a one-fits-all approach to evaluation, and instead to evaluate plagiarism detectors step-wise. Our focus is on the candidate retrieval task and the detailed comparison task, we devised new evaluation frameworks for each of them. In the following two sections, we detail the evaluations of both tasks.

[1] Another approach to detect plagiarism is called *intrinsic* plagiarism detection, where detectors are given only a suspicious document and supposed to identify text passages in them which deviate in their style from the remainder of the document. In this year's competition, we focus on external plagiarism detection, while intrinsic plagiarism detection has been evaluated in a related sub-task within PAN's authorship identification task [15].

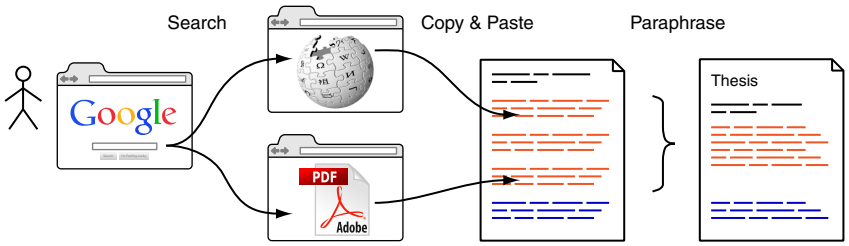

Figure 2. The basic steps of plagiarizing from the web [21].

2 Candidate Retrieval Evaluation

The web has become one of the most common sources for plagiarism and text reuse. Presumably, humans follow the three steps outlined in Figure 2 when reusing or plagiarizing text from the web: starting with a search for appropriate sources on a given topic, text is copied from them, and afterwards possibly modified to the extent that the author believes it may not be easily detected anymore. Unsurprisingly, manufacturers of commercial plagiarism detectors as well as researchers working on this subject frequently claim their systems to be searching the web or at least be scalable to its size. However, there is hardly any evidence to substantiate these claims. Neither have commercial plagiarism detectors been found to reliably identify plagiarism from the web,[2] nor have many researchers presented convincing evaluations of their prototypes [24]. In this year's competition, we address this issue for the first time by developing a corpus specifically suited for candidate retrieval from the web.

Related Work. The previous PAN corpora[3] marked a major step forward in evaluating plagiarism detectors [24]. But these corpora have a number of shortcomings that render them less realistic compared to real plagiarism from the web:

- All plagiarism cases were generated by randomly selecting text passages from source documents and inserting them at random positions in another host document. This way, the "plagiarized" passages do not match the topic of the host document.
- The majority of the plagiarized passages were modified to emulate plagiarist behavior. However, the strategies applied are, again, basically random (i.e., shuffling, replacing, inserting, or deleting words randomly). An effort was made to avoid non-readable text, yet none of it bears any semantics.
- The corpus documents are parts of books from the Project Gutenberg.[4] Many of these books are rather old, whereas today the web is the predominant source for plagiarists.

[2] http://plagiat.htw-berlin.de/software-en/2010-2
[3] http://www.webis.de/research/corpora
[4] http://www.gutenberg.org

With respect to the second issue, about 4000 plagiarized passages were rewritten manually via crowdsourcing on Amazon's Mechanical Turk. Though the obtained results are of a high quality, an analysis of topic drift in a suspicious document still reveals a plagiarized passage more easily than actually searching its original source [25]. While neither of these issues is entirely detrimental to evaluation, it becomes clear that there are limits to constructing plagiarism corpora automatically.

Besides the PAN corpora, there is only one other that explicitly comprises plagiarized text. The Clough09 corpus consists of 57 short answers to one of 5 computer science questions which were plagiarized from a given Wikipedia article [2]. While the text was genuinely written by a small number of volunteer students, the choice of topics is very narrow, and text lengths range only from 200 to 300 words, which is hardly more than 2-3 paragraphs. Also, the sources to plagiarize were given up front so that there is no data on retrieving them.

A research field closely related to plagiarism detection is that of text reuse. Text reuse and plagiarism are in fact two of a kind [21]: text reuse may appear in the forms of citations, boilerplate text, translations, paraphrases, and summaries, whereas all of them may be considered plagiarism under certain circumstances. Text reuse is more general than plagiarism but the latter has received a lot more interest in terms of publications. For the study of text reuse, there is the Meter corpus, which comprises 445 cases of text reuse among 1716 news articles [3]. The text reuse cases found in the corpus are realistic for the news domain, however, they have not been created by the reuse process outlined in Figure 2.

Contributions. Altogether, we lack a realistic, large-scale evaluation corpus for candidate retrieval from the web. For this year's competition, we hence decided to construct a new corpus comprising long, manually written documents. The corpus construction, for the first time, emulates the entire process of plagiarizing or reusing text shown in Figure 2, both at scale and in a controlled environment. The corpus comprises a number of features that set it apart from previous ones: (1) the topics of each plagiarized document in the corpus are derived from the topics of the TREC Web Track, and sources have been retrieved from the ClueWeb corpus. (2) The search for sources is logged, including click-through and browsing data. (3) A fine-grained edit history has been recorded for each plagiarized document. (4) A total of 300 plagiarized documents were produced, most of them 5000 words long, while ensuring diversity via crowdsourcing.

2.1 Corpus Construction Tools: TREC Topics, ClueWeb, ChatNoir, Web Editor

This section details the ingredients that went into the development of our new corpus and the constraints that were to be met. Generally, when constructing a new corpus, a good starting point is a clear understanding of a-priori required resources. With regard to our corpus, we distinguish three categories: data, technologies, and human resources. For each of these, a number of constraints as well as desirable properties and characteristics were identified.

Data. Two pieces of data were required for corpus construction: a set of topics about which documents were to be written, and a corpus of web pages to be used as sources

for plagiarism. Coming up with a variety of topics to write about is certainly not a big problem. However, we specifically sought not to reinvent the wheel, and when looking for sources of topics, TREC can hardly go unnoticed. After reviewing the topics used for the TREC Web Track,[5] we found them amenable for our purpose. Hence, our topics are derived from TREC topics, rephrasing them so that one is asked to write a text instead of searching for relevant web pages. Yet, writing a text on a given topic may still include the task of searching for relevant web pages. For example, below is a quote of topic 001 of the TREC Web Track 2009:

> *Query*: obama family tree
>
> *Description*: Find information on President Barack Obama's family history, including genealogy, national origins, places and dates of birth, etc.
>
> *Sub-topic 1*: Find the TIME magazine photo essay "Barack Obama's Family Tree."
>
> *Sub-topic 2*: Where did Barack Obama's parents and grandparents come from?
>
> *Sub-topic 3*: Find biographical information on Barack Obama's mother.

This topic has been rephrased as follows:

> **Obama's family**. Write about President Barack Obama's family history, including genealogy, national origins, places and dates of birth, etc. Where did Barack Obama's parents and grandparents come from? Also include a brief biography of Obama's mother.

All except one sub-topic could be preserved, whereas sub-topic 1 was considered too specific to be of real use, especially since a photo essay does not contain a lot of text. Two groups of topics are among the TREC Web track topics, namely faceted and ambiguous ones. The former were easier to be translated into essay topics, whereas for the latter, we typically chose one of the available ambiguities.

With regard to the corpus of web pages used as sources for plagiarism, one of the top requirements was a huge size in order to foreclose exhaustive comparisons of suspicious documents to all the documents in the corpus. The corpus should also be as representative as possible compared to the entire web. To date, one of the most representative web crawls available to researchers is the ClueWeb corpus.[6] The corpus consists of more than 1 billion documents from 10 languages which amount to 25 terabytes of data. It has become a widely accepted resource, being used to evaluate the retrieval performance of search engines in the course of the TREC Web Track 2009–2011. Other publicly available corpora include the DOTGOV crawl and the WT10G crawl, which were previously used in TREC, as well as the crawl released by the Commoncrawl initiative, which is 5 times larger than the ClueWeb corpus.[7] However, since the ClueWeb corpus is still the de-facto standard evaluation corpus, we decided to stick with it. Consequently, we have limited our choice of topics to those used within the TREC Web Tracks 2009–2011.

[5] http://trec.nist.gov/tracks.html
[6] http://lemurproject.org/clueweb09.php
[7] http://commoncrawl.org

Technologies. Three pieces of technology were required for corpus construction: a search engine for the aforementioned web corpus, a web service that serves web pages from the web corpus on demand, and a text editor with which the plagiarized documents were to be written. Unfortunately, neither of these technologies could be obtained off the shelf, since we required full access to them in order to closely track our human subjects during corpus construction to make measurements and collect data.

For the sake of realism, we expected the search engine used for corpus construction to resemble commercial ones as close as possible, so that our human subjects behave naturally and similar to "real plagiarists." This requires not only a state-of-the-art web retrieval model, but also an intuitive interface as well as fast retrieval. Until now, there has been no search engine publicly available to researchers that combines all of these properties. However, our work on this corpus construction project coincided with the development of a new research search engine for the ClueWeb at our site: ChatNoir [26].

The ChatNoir search engine is based on the classic BM25F retrieval model [27], using the anchor text list provided by the University of Twente [13], the PageRanks provided by the Carnegie Mellon University,[8] and the spam rank list provided by the University of Waterloo [4]. ChatNoir also incorporates an approximate proximity feature with variable-width buckets as described by Elsayed et al. [5]: the text body of each document is divided into 64 buckets such that neighboring buckets have a half-bucket overlap. For each keyword, not the exact position is stored in a 1-gram index but its occurrence in the individual buckets is indicated via a bit flag. Hence, for each keyword in a document, a 64-bit vector stores whether it occurs in one of the 64 buckets. This retrieval model is of course not as mature as those of commercial search engines; yet it combines some of the most widely accepted approaches in information retrieval. In addition to the retrieval model, ChatNoir also implements two search facets relevant to those who plagiarize from the web: text readability scoring, and long text search. The former facet, similar to that offered by Google, scores the readability of the text found on every web page using the well-known Flesh-Kincaid grade level formula. It estimates the number of years one should have visited school in order to understand a given text. This number is then mapped onto the three fuzzy categories „simple," „intermediate," and „expert." The long text search facet filters search results which do not contain at least one continuous paragraph of text longer than a threshold of 300 words. The facets can be combined with each other. ChatNoir indexes 1-grams, 2-grams, and 3-grams, and the implementation of the underlying inverted index has been optimized to guarantee fast response times. It runs on a cluster of 10 standard quad-core PCs and 2 eight-core rack servers. Short to medium length queries are answered between 1 and 5 seconds. The web interface of ChatNoir resembles that of commercial search engines in terms of search result presentation and usability.[9]

When a user of our search engine is presented with her search results and then clicks on one of them, she is not redirected into the real web but the ClueWeb instead. Although the ClueWeb provides the original URLs from which the web pages have been obtained, many are dead or have been updated since the corpus was crawled. Moreover, in order to create traceable plagiarism cases, the plagiarized texts should come from

[8] http://boston.lti.cs.cmu.edu/clueweb09/wiki/tiki-index.php?page=PageRank

[9] http://chatnoir.webis.de

a web page which is available in the ClueWeb. We hence have set up a web service that serves web pages from the ClueWeb on demand: when accessing a web page, it is pre-processed before being shipped, removing all kinds of automatic referrers, and replacing all links that point into the real web with links onto their corresponding web page inside the ClueWeb. This way, the ClueWeb may be browsed as if surfing the real web, whereas it becomes possible to precisely track a user's movements. The ClueWeb itself is stored in the HDFS of our 40 node Hadoop cluster, and each web page is fetched directly from there with latencies around 200ms.

Next to the search engine and the ClueWeb access service, the third technology required for corpus construction must strike as a simple one: a basic text editor. However, in order to properly trace how a plagiarism case was constructed, the human subjects who plagiarized on our behalf must be minutely tracked while modifying a piece of text copied from a web page that has been retrieved with our search engine. Furthermore, the editor to be used should allow for remote work since we did not know in advance who was going to plagiarize for us. Looking at the available alternatives (Microsoft Word, OpenOffice, Google Docs, Zoho, rich text editors, etc.) we decided to go with the simplest option of web-based rich text editors. They offer maximal flexibility while being simple to use: using a well-known web toolkit, a web application has been developed that features a rich text editor alongside a set of instructions. The editor provides an autosave feature that sends the current state of a text to our servers, every time the user stops typing for more than 300ms. On the server side, each new revision of a given text is stored into a Git repository. This way, a detailed revision history is recorded which tracks the edits made to a text in much finer detail than, for instance, those of Wikipedia articles. Moreover, the editor enables its users to track the sources of text copied into it by allowing for it to be colored, and each color to be linked to the web page the text came from (cf. Figure 3, right column). This manual effort of immediate annotation was required from our human subjects in order to ease post-processing and quality control.

Human Resources. Having a working, controlled web search environment at hand, an open question was who to pick as volunteer plagiarists. Short of real ones, we employ people who act as plagiarists. Since the target language of the corpus is English, we require them to be fluent English speakers and preferably have some writing experience. We assume that (semi-)professional writers are not only faster but also have at their disposal more diverse ways of modifying a text so as to vary the outcome. We have made use of crowdsourcing platforms to hire writers. Crowdsourcing has quickly become one of the cornerstones of constructing evaluation corpora. This is especially true for paid crowdsourcing via Amazon's Mechanical Turk [1]. This platform is frequently used by researchers to annotate and collect data. On the upside, it offers a flexible interface, a large workforce, and very low costs per unit. Many report on collecting thousands of annotations for just a couple of dollars. On the downside, however, scammers constantly submit fake results in order to get paid without actually working, so that quality control is one of the main obstacles to using Mechanical Turk. Moreover, the workforce has very diverse skills and knowledge so that task design and simplicity of task description have severe impact on result quality. Since our task is hardly simple enough for

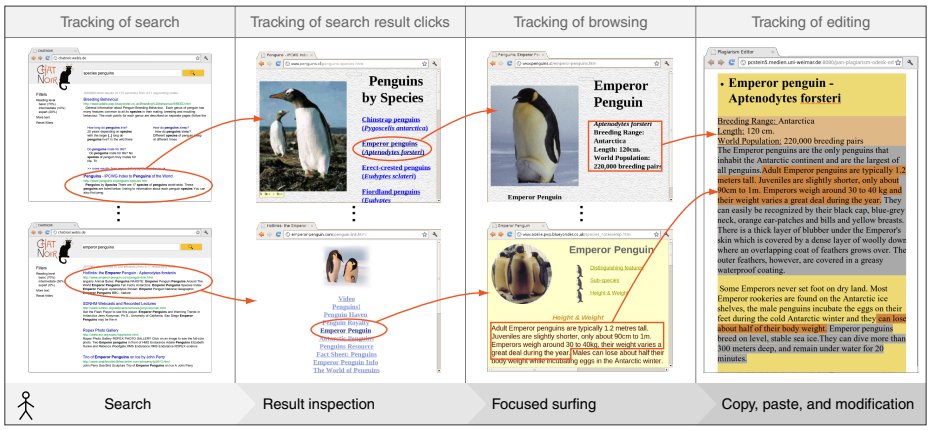

Figure 3. Corpus example (topic 058): plagiarized passages can be traced back to their source web pages and the queries that retrieved them.

Mechanical Turk, we resort to oDesk instead.[10] At oDesk, wages are higher, whereas scamming is much lower because of more advanced rating features for workers and employers. Finally, at our site, two post docs worked part time to supervise the oDesk workers and check their results, whereas the technologies were kept running by three undergrad students.

2.2 Corpus Construction

To construct our corpus, we hired 18 (self-proclaimed) professional writers at oDesk, coming from various countries and backgrounds. A job advertisement was posted at oDesk to which more than half of the workforce replied, whereas the other authors were sought out manually. Employment criteria included a reasonable application to the job ad, a reasonable number of previously billed hours in other writing appointments, and, for budgetary reasons, an hourly rate below 25 US$. Nevertheless, a few writers above that rate were hired to ensure diversity. The hourly rates ranged from 3.33 US$ to 34 US$ with an average of 12.43 US$. Before hiring writers at oDesk, we recruited 10 colleagues and friends to test the environment by producing a plagiarized document each, so that a total of 28 different writers were involved in corpus construction.

Plagiarized documents were produced in two batches of 150 documents each, so that each of the aforementioned TREC topics has been written about twice. No writer was allowed to pick the same topic twice. There was a key difference between the first and the second batch which relates to the retrieval model employed for searching source documents in the ClueWeb. Given the fact that the ChatNoir search engine implements only a single, basic retrieval model, our setup bears the risk of a systematic error if writers use only our search engine to search for source documents on a given topic

[10] http://www.odesk.com

(e.g., implementation errors, poor retrieval performance, etc.). Hence, we decided to circumvent this risk by reusing TREC qrels (documents retrieved by participants of the TREC Web Tracks and judged for relevance by the TREC organizers) which achieved at least the relevance level 2 of 3 on a given topic. For each topic in the first batch, writers were asked to plagiarize from as many of the topic-specific qrels as they wished, without using ChatNoir. In the second batch, no source documents were given and writers were asked to use ChatNoir to retrieve sources matching a given topic. This way, the plagiarism in the documents from the first batch stems from source documents that were retrieved by the union of all retrieval models that took part in the TREC Web Tracks, whereas the second batch is based on ChatNoir's retrieval model only.

Each writer worked on one topic at a time while being allowed to choose from the remaining topics. They were asked to write documents of at least 5000 words length, which sometimes was not possible due to lack of source material. Besides plagiarized text passages, they were also asked to genuinely write text of their own. In this connection, we observed that, once plagiarism is allowed, writers became reluctant to write genuine text. The number of possible sources was limited to up to 30 for the first batch and increased to up to 75 during the second batch. Throughout corpus construction, a number of measurements were made so that substantial meta data can be provided in addition to each plagiarized document (Figure 3 shows an example of this data):

- Plagiarized document
- Set of source documents plagiarized.
- Annotations of passages plagiarized from the sources.
- Log of queries posed by the writer while writing the text.
- Search results for each query.
- Click-through data for each query.
- Browsing data of links clicked within ClueWeb documents.
- Edit history of the document.

Finally, each plagiarized document, along with its meta data, is post-processed to correct errors and to convert it into a machine-readable format that suits the needs of evaluating plagiarism detectors. The post-processing step includes double-checking whether the source documents are all contained in the ClueWeb, and whether the annotation of text passages to source documents is correct. Moreover, because of the rather crude HTML code that is generated by today's rich text editors, each plagiarized document must be manually annotated in order to mark the beginning and end of each plagiarized text passage. At the time of writing, post-processing the plagiarized documents constructed for our corpus was still a work in progress.

2.3 Evaluation with the ChatNoir Search Engine

Since our new corpus uses the ClueWeb as the collection of source documents, it is this collection that must be input to a plagiarism detector implementing the retrieval process shown in Figure 1. Given the prize and size of the ClueWeb, however, we could hardly expect participants to first buy a license and then index the ClueWeb in the three months of the competition's training phase. In the 2009 plagiarism detection competition, our

first plagiarism corpus of more than 40 000 text documents already posed a challenge to participants, so that handling the 0.5 billion English web pages of the ClueWeb would have been an insurmountable obstacle to most prospective participants. It is hence that ChatNoir served two purposes in our evaluation: its user interface was used by our human plagiarists to search for source documents, and its API was used by participants to develop retrieval algorithms that rediscover the sources of a suspicious document. This is a realistic approach, since it is likely that real plagiarists use the same search engine as commercial plagiarism detectors.

Evaluation Corpus. The competition was divided into two phases, a training phase and a test phase. For both phases, a subset of our corpus was released. The training corpus consisted of 8 plagiarized documents including annotations that indicated which text passage was plagiarized from which ClueWeb document, whereas the test corpus (consisting of 32 documents) naturally did not contain such annotations. Note that the evaluation corpora used in this year's candidate retrieval task did not include all plagiarized documents created, but only a subset of them. There are two reasons for this: at the time of release of the training and test corpora, not all documents were finished, and post-processing the raw documents obtained from the oDesk writers was too time-consuming so that no further documents could be added in time for the release.

Performance Measures. The performance of a candidate retrieval algorithm depends on what is its desired behavior. The candidate retrieval step of plagiarism detection is supposed to filter the collection of potential source documents and to output only the candidate documents that are likely sources for plagiarism with respect to a given suspicious document. Ideally, a candidate retrieval algorithm would retrieve exactly the documents used as sources by a plagiarist, and none other. This way, the subsequent steps in plagiarism detection would have to deal only with documents that are worthwhile processing. Furthermore, when considering that the document collection to be searched by a candidate retrieval algorithm may be the entire web, it becomes clear that existing retrieval models, search engines, and search infrastructures should be reused for this task. Building one's own search engine is out of bounds for many, whereas access to existing search engines is usually not free of charge (both in terms of cost per query and retrieval time). Therefore, an ideal candidate retrieval algorithm also minimizes the amount of queries posed to a search engine in order to retrieve candidate documents. With these considerations in mind, we measure candidate retrieval performance for each suspicious document in the test corpus using the following five scores:

1. Number of queries submitted.
2. Number of web pages downloaded.
3. Precision and recall of web pages downloaded regarding the actual sources.
4. Number of queries until the first actual source is found.
5. Number of downloads until the first actual source is downloaded.

Measures 1–3 capture a candidate retrieval algorithm's overall behavior and measures 4–5 assess the time to first detection. Note in this connection, that neither of these measures captures the quality of extracting plagiarized passages from suspicious documents since this is supposed to happen within the detailed comparison of candidate documents with a suspicious document.

2.4 Survey of Retrieval Approaches

Five of the 15 participants submitted runs for the candidate retrieval task, four of whom also submitted a notebook describing their approach. An analysis of these notebooks reveals a number of building blocks that were commonly used to build candidate retrieval algorithms: (1) chunking, (2) keyphrase extraction, (3) query formulation, (4) search control, and (5) download filtering. In what follows, we describe them 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 25-line chunking [7], 4-sentence chunking [14], paragraph chunking [17], and no chunking [32]. Neither of the participants mention the extent to which the chunks produced by their strategies overlap.

Keyphrase Extraction. Given a chunk (or the entire suspicious document), 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 candidate retrieval algorithm since the decisions made here directly affect overall performance: the fewer keywords are extracted, the better the choice must be or recall is irrevocably lost.

Since ChatNoir, during this year's test phase, unfortunately still lacked phrasal search, participants resorted to extracting keywords. Kong et al. [17] rank a chunk's words by their $tf \cdot idf$ values, where a word's document frequency is presumably obtained from a large external document collection, and then extract the top 10 most discriminative words. Jayapal [14] uses the first 10 words of a chunk whose part-of-speech are either noun, pronoun, verb, or adjective. Gillam et al. [7] rank a chunk's words using a so-called "enhanced weirdness" score which measures the likelihood of a word not appearing in general text, then re-rank the top 10 words by their frequency, and return the top most frequent term plus the 4 words succeeding it. The keyphrase extractor of Suchomel et al. [32] goes beyond the aforementioned ones in that it employs not one but three strategies at the same time: (1) a chunk's words are ranked by their $tf \cdot idf$ values, where a word's document frequency is obtained from an English web corpus, and then extract the top most discriminative words that exceed a threshold. (2) The suspicious document is chunked in 45-word chunks (40 word overlap), then chunks are picked whose vocabulary richness is significantly higher than that of neighboring chunks, from each of which the first sentence longer than 8 words (excluding stop words) is selected and the first 6 words (excluding stop words) are extracted. (3) Headings in the suspicious document are identified and extracted (excluding stop words).

A key insight from reviewing the participants approaches is that Suchomel et al.'s strategy of combining different ideas to extract keyphrases is probably superior to the

one-fits-all approach of the other participants. This way, just as with chunking, the risk of algorithm error is further diminished and it becomes possible to exploit different sources of information that complement each other. A minor criticism is that only custom-made keyphrase extractors were used, and hardly any reference was made to the extensive literature on the subject. While it may be necessary to tailor keyphrase extraction methods to the task of candidate retrieval, existing work should not be entirely neglected.

Query Formulation. Given sets of keywords extracted from chunks, queries are formulated which are tailored to the API of the search engine used. Rationale for this is to adhere to restrictions imposed by the search engine and to exploit search features that go beyond basic keyword search. The maximum number of search terms enforced by ChatNoir is 10 keywords per query. All participants except Jayapal [14], who formulated 10-word queries, formulated 5-to-6-word queries, using the word ranking of their respective keyword extractor. The reason for limiting query length was to avoid overspecific queries. Interestingly, the participants formulate non-overlapping queries (i.e., they do not use the same keyword in more than one query), in contrast to previous candidate retrieval strategies in the literature [12]. Also note that non of the participants made use of the search facets offered by ChatNoir, namely the facet to search for web pages of at least 300 words of text, and the facet to filter search results by readability.

Search Control. Given a set of queries, the search controller schedules their submission to the search engine and directs the download of search results. Rationale for this is to dynamically adjust the search based on the results of each query, which may include dropping queries, reformulating existing ones, or formulating new ones based on the relevance feedback obtained from search results. Only Suchomel et al. [32] implemented a search controller, whereas all other participants simply submitted all queries to the search engine, downloading all, or only the top n results. Suchomel et al.'s search controller schedules queries dependent on the keyphrase extractor which extracted their words: the order of precedence corresponds to the order in which they have been explained above. Then, for each search result obtained from submitting one open query, it is checked whether its snippet is contained in the suspicious document. If so, the document is downloaded and subject to detailed comparison to the suspicious document. In case, a plagiarized passage is discovered, all queries whose keywords originate from that portion of the suspicious document are discarded from the list of open queries. No attempts are made at reformulating existing queries or formulating new ones based on the documents downloaded.

Download Filtering. Given a set of downloaded documents, a download filter removes all documents that are probably not worthwhile being compared in detail with the suspicious document. Rationale for this is to further reduce the set of candidates and to save invocations of the subsequent detailed comparison step. All except one participant skipped this step and proceeded to detailed comparison directly. Jayapal [14] computes the 5-gram Jaccard similarity of each downloaded document to the suspicious document and discards documents that do not exceed a similarity threshold.

Table 1. Performances on the candidate retrieval subtask. Values are averaged over the 32 suspicious documents from the test corpus. The top half of the table shows performances when interpreting near-duplicates of the actual source documents as true positives; the bottom half of the table shows performances without considering near-duplicates true positives.

Team	Total Workload		Time to 1st Detection		No Detection	Reported Sources		Downloaded Sources		Retrieved Sources	
	Queries	Downloads	Queries	Downloads		Precision	Recall	Precision	Recall	Precision	Recall
Gillam	63.44	527.41	4.47	25.88	**1**	0.6266	0.2493	0.0182	**0.5567**	0.0182	**0.5567**
Jayapal	67.06	173.47	8.78	13.50	**1**	**0.6582**	**0.2775**	0.0709	0.4342	**0.0698**	0.4342
Kong	551.06	326.66	80.59	27.47	2	0.5720	0.2351	0.0178	0.3742	0.0141	0.3788
Palkovskii	63.13	1026.72	27.28	318.94	6	0.4349	0.1203	0.0025	0.2133	0.0024	0.2133
Suchomel	**12.56**	**95.41**	**1.53**	**6.28**	2	0.5177	0.2087	**0.0813**	0.3513	0.0094	0.4519
Gillam	63.44	527.41	52.38	445.25	22	0.0310	0.0414	0.0016	0.0526	0.0019	0.0526
Jayapal	67.06	173.47	39.00	115.13	16	0.0328	0.0394	0.0079	0.0994	0.0108	0.0994
Kong	551.06	326.66	440.59	274.06	21	0.0280	0.0458	0.0019	0.0391	0.0015	0.0435
Palkovskii	63.13	1026.72	54.88	881.34	25	0.0246	0.0286	0.0002	0.0286	0.0002	0.0364
Suchomel	12.56	95.41	11.16	93.72	30	0.0208	0.0124	0.0007	0.0124	0.0003	0.0208

2.5 Evaluation Results

Table 1 shows averaged performances of the five candidate retrieval algorithms over the 32 plagiarized documents that formed the test corpus. While computing the performances, we encountered a problem that required (semi-)automatic post-processing of the submitted runs, namely that many web pages in the ClueWeb are near-duplicates of one another. Although the plagiarized passages in the corpus documents were lifted from exactly one web page, it is very well possible that a candidate retrieval algorithm retrieves a different page with the same contents. In such cases, the candidate algorithm is not in error and hence its performance should not be discounted. To alleviate this issue we compared all documents retrieved by the candidate retrieval algorithms in turn to each of the 362 sources for plagiarism in the test corpus in order to identify near-duplicate web pages. Here, two web pages are considered near-duplicates if their cosine similarity under a tf-weighted 1-gram vector space model is higher than 0.85 and higher than 0.7 under a tf-weighted 3-gram vector space model. A number of spot checks were made to ensure a low rate of false positives, however, since some plagiarized passages in our corpus are rather short, there is the possibility that some of the documents retrieved were falsely considered not being duplicates of the actual source documents. The top half of Table 1 shows performances when treating near-duplicates as positive source documents, and the bottom half shows performances when counting only detections of actual source documents. The differences are profound, indicating that most candidate retrieval algorithms mainly retrieved near-duplicates of the plagiarized documents' actual sources from the ClueWeb.

The column group "Reported Sources" shows precision and recall of the sets of candidate documents that have been submitted as runs by the participants, whereas the performances shown in all other columns and column groups are based on access log data recorded at our site. To facilitate these measurements, each participant's candidate

retrieval algorithm was assigned a unique access token and an exclusive access time period to ChatNoir in order to process the test corpus. All other access methods were disabled during the test phase. Furthermore, participants were asked to indicate during access which plagiarized document of the test corpus is currently processed by their algorithm. This enabled us to record detailed information about queries submitted, search results obtained, and downloads made.

The three column groups "Retrieved Sources," "Downloaded Sources," and "Reported Sources" measure performance at specific points of executing a candidate retrieval algorithm: after all queries have been submitted, after selected search results have been downloaded, and after the final decision is made as to which downloaded documents are considered candidates. Hence, the set of retrieved documents equals or subsumes the set of downloaded ones, which in turn equals or subsumes the set of reported documents. It can be seen that noteworthy precision is only achieved regarding reported sources, whereas recall is at times more than double as high on downloaded and retrieved sources compared to reported ones. This indicates that all candidate retrieval algorithms trade a lot of recall for precision during download filtering. The best performing approach regarding reported sources is that of Jayapal [14], while coming in second to Gillam et al. [7] regarding downloaded and retrieved sources.

The three remaining column groups indicate the total workload and the time to first detection of each candidate retrieval algorithm, given in absolute terms of queries submitted and downloads made. The column "No Detection" indicates for how many plagiarized documents no true positive detection was made, which shows very low values in the top half of the table. Otherwise, the average number of queries and downloads ranges from tens to thousands. The best performing approach by a long margin is that of Suchomel et al. [32]. Here, their search controller pays off, while presumably the combination of three keyword extractors maintains high precision and recall regarding reported sources: despite their low numbers of queries and downloads, a remarkably competitive precision and recall is achieved. Clearly, this approach is the most cost-effective one. The approach of Jayapal, however, requires only six times as many queries overall and less than twice as many downloads.

Discussion. The obtained results validate our setup: the search engine apparently retrieves web pages that are useful for both humans and machines. Especially, it is possible for a machine plagiarism detector to trace the steps of a human plagiarist. That said, there is still a lot of room for improvement concerning the tools used within our new evaluation framework and the participants' candidate retrieval approaches. With regard to our evaluation framework, ChatNoir is still a single point of failure, and improvements to its retrieval model directly affect participants' performances. Moreover, the performance measures used above require further research: from the evaluation results, it can already be concluded that candidate retrieval is a recall-oriented task, whereas precision mainly becomes a measure of runtime of subsequent steps. Finally, the participants' approaches are still somewhat immature (as can be expected when introducing a new evaluation framework [22]). Their detection quality is rather low, which limits the overall detection performance of a web plagiarism detector. Also, the research presented is often not well-founded in the literature.

3 Detailed Comparison Evaluation

After a set of candidate source documents has been retrieved for a suspicious document, the follow-up task of an external plagiarism detection pipeline is to analyze in detail whether the suspicious document in fact contains plagiarized passages from these sources (cf. Figure 1), and to extract them with high accuracy. The major difference of this year's detailed comparison evaluation compared to previous years is that we asked participants to submit their detection software instead of just detection results on a given data set, which has a couple of advantages:

- *Runtime Analysis.* Software submissions of detailed comparison approaches, allows for measuring and comparing runtime characteristics, since, for commercial plagiarism detectors, efficiency is just as important as detection effectiveness.
- *Real Plagiarism Cases.* Real plagiarism cases are the ultimate resource to evaluate plagiarism detectors. In previous competitions, privacy concerns prevented us from using them in our evaluation, since these cases would have to be released to the participants. Having the software at our site, this year, a small set of real plagiarism cases was incorporated into the test collection for the first time.
- *Continuous Evaluation.* In the course of the previous competitions, the employed corpora have changed considerably (improving their quality in each iteration). Hence, it has become difficult to compare detection performances across years and thus, it is hard to tell to which extent the approaches have improved over time. For instance, in 2011, the best overall detection performance was, in absolute terms, below that of 2010, whereas the evaluation corpus used in 2011 was a lot more difficult than before. With software submissions, it is now possible to continuously compare approaches from multiple years whenever new collections are released, given the consent of the respective authors.

This year, eleven teams submitted software, which is comparable to last year's number of participants. Software submissions are hence no big obstacle to participants and should be pursued further in the future.

3.1 Evaluation with the Experimentation Platform TIRA

Besides the aforementioned advantages of software submissions, there are also disadvantages, for instance, the non-negligible amount of extra work on the organizer's side. One possibility to reduce the complexity of the evaluation task is to constrain participants to a certain operating system and programming language. However, due to the diverse coding preferences of software developers and the fact that a mismatch of our constraints to their preferences would foreclose reusing existing prior work, it is futile to find a platform that satisfies all participants. We hence decided to allow for detailed comparison software of various kinds, and use the experimentation platform TIRA [10] to keep the organizational overhead moderate. This way, we kept the criteria to be met by participants' software at a minimum: it must be executable on either a Windows or a Linux platform, it must accept two input parameters (the location of the suspicious document and that of a candidate document) and return detection results in a pre-specified

XML format, and it must be accompanied by a comprehensive installation manual. Furthermore, TIRA provides a set of features that facilitate our work [9]:

- *Experiments as a Service.* TIRA creates a web service for every deployed program. The web service can be accessed remotely using a web browser. It allows for the convenient execution of a program with individual parameter settings as well as for the retrieval of already computed experiment results.
- *System Independence.* Shipped as a Java program, TIRA can be instantiated on both Windows and Linux platforms. Deployed programs are executed from a command shell of the underlying operating system, rendering program execution independent of a specific programming language or middleware.
- *Peer To Peer Networking.* The possibility to join TIRA instances on different machines to form a TIRA *network* allows to control programs running on different operating systems from a single web page.
- *Multivalued Configurations.* TIRA provides an intuitive mechanism for specifying multiple values for program parameters. This way, a large series of program runs can be initiated by a single run configuration.
- *Distributed Execution.* The scheduling mechanism of TIRA supports the efficient execution of programs in different threads and across multiple machines.
- *Result Retrieval.* TIRA provides means for storing, indexing, and retrieving experiment results.

TIRA was employed in the training phase as well as the test phase of this year's competition. In the training phase, participants were given the possibility to upload detection results which they obtained on the released training corpus. As a response, TIRA returned the performance values for the submission. Since the performance values have been publicly visible, our idea was to use TIRA as an online leader board for the early phase of the competition. However, the service has not been adopted until the last week before the run submission deadline [8]. For the test phase, a Windows 7 and an Ubuntu 12.04 virtual machine was set up, each running on a computer with 70 GB RAM and two quad-core Intel Xeon E552 CPUs, and TIRA as well as all program submissions were installed. For one submission, we experienced unsolvable runtime problems and, in mutual agreement with the authors, omitted this submission from the evaluation. The two virtual machines were accessible from the local network, and linked to a third TIRA instance which served as the control instance. From this control instance the evaluation of all submissions on all of the sub-corpora of the test corpus was managed. Computing the results for all submissions took two days.

Evaluation Corpus. As an evaluation corpus, we employed the corpus construction process used in previous competitions. Based on the books of Project Gutenberg, we extracted passages from one subset of the books (source documents), obfuscated them, and inserted them as plagiarized passages into a different subset of books (suspicious documents). For the obfuscation process, the four strategies No Obfuscation, Artificial (Low), Artificial (High), and Manual Paraphrasing were used. In addition, the eval-

Table 2. Corpus statistics of the PAN 2012 detailed comparison test corpus.

Evaluation Corpus Statistics		
Sub-Corpus	Number of Cases	Avg. Cosine Similarity
Real Cases	33	0.161
Simulated	500	0.364
Translation ($\{$de, es$\} \rightarrow$ en)	500	0.018
Artificial (High)	500	0.392
Artificial (Low)	500	0.455
No Obfuscation	500	0.560
No Plagiarism	500	0.431
Overall / Averaged	3033	0.369

uation corpus contains documents without any plagiarized passages. A more detailed description of the obfuscation strategies can be found in [21].

Similar to previous years, we also constructed a set of cross-language plagiarism cases. This time, however, we did not apply Google Translate as it was observed that such a construction strategy is biased towards detailed comparison approaches which themselves use Google Translate to detect cross-language plagiarism. Instead, we completely revised the cross-language sub-corpus and created the cases based on the multilingual Europarl corpus [16]. Starting from a non-English source document in the corpus, we first eliminated all paragraphs not coming from the document's main speaker, selected a passage to be plagiarized, extracted the corresponding passage from the English version of the source document, and inserted it into a Gutenberg book. Another improvement over previous years is that document similarity was considered when choosing a suspicious document for a given source document. Especially pairs of suspicious and source document pairs without plagiarized passages or unobfuscated ones, we intended to make the detection task more challenging and realistic by selecting books with a similar vocabulary. Similarities were computed under a tf-weighted vector space model, employing stop word removal, and the cosine similarity metric. The average similarities for each of the sub-corpora and their respective sizes are given in Table 2. Finally, the 33 real plagiarism cases used this year were obtained by manually collecting them from the Internet. Each case consists of passages of no more than 75-150 words, and they were found not to be embedded in host documents. We hence resorted to embedding them, automatically, into topically related host documents sampled from Wikipedia. Of 10 000 Wikipedia articles, for each plagiarism case, two different host documents were chosen which are similar to the plagiarized passage under a $tf \cdot idf$-weighted vector space model and the cosine similarity. Then, the plagiarism case's plagiarized and source passages were inserted at randomly selected positions of the two documents, each in one of them.

Performance Measures. To assess the performance of the submitted detailed comparison approaches, we employed the performance measures used in previous competitions.

For this paper to be self-contained, we summarize the definition found in [24]: let S denote the set of plagiarism cases in the corpus, and let R denote the set of detections reported by a plagiarism detector for the suspicious documents. To simplify notation, a plagiarism case $s = \langle s_{\text{plg}}, d_{\text{plg}}, s_{\text{src}}, d_{\text{src}} \rangle$, $s \in S$, is represented as a set $\mathbf{s}$ of references to the characters of d_{plg} and d_{src}, specifying the passages s_{plg} and s_{src}. Likewise, a plagiarism detection $r \in R$ is represented as $\mathbf{r}$. Based on this notation, precision and recall of R under S can be measured as follows:

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

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

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

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

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

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

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

Analysis of the $plagdet$ Score. The $plagdet$ score has been criticized to put too much weight on granularity instead of precision and recall, thus ranking plagiarism detectors differently than a human would. We stress that $plagdet$ was not meant to be the single yardstick of evaluation and that—just as with the F_α-Measure—one should always look at individual performance measures to judge an algorithm's performance. Though a number of experiments were conducted to verify the score, we did not continue research in this direction, until now. Inspired by the experiments of Grozea and Popescu [11], we now close this gap and shed further light on how $plagdet$ ranks plagiarism detectors.

Grozea and Popescu have identified a situation in which $plagdet$ ranks two hypothetical plagiarism detectors in a possibly counterintuitive way. The situation involves one detector A with the performance characteristic $\langle prec_A = 1,\ rec_A = 0.5,\ gran_A = 1 \rangle$ and another detector B with the characteristic $\langle 1, 1, 2 \rangle$.[11] In plain terms, given a single plagiarism case, detector A would detect half of it in one piece, whereas detector B would detect the entire plagiarism case in two pieces. The $plagdet$

[11] In Grozea and Popescu's example, B's performance characteristic is first defined as $\langle 1, 1, 3 \rangle$, but later the granularity is changed to 2. For consistency, we set granularity to 2 right away.

score of detector A is 0.67, and that of detector B is 0.63, so that A is ranked higher than B. It is debatable which of the two plagiarism detectors humans prefer in general, the one that detects a part of each plagiarism case in one piece, or the one that detects entire plagiarism cases in many pieces. Incidentally, the introduction of the granularity measure forced developers of plagiarism detectors for the first time to actually care about how often a single plagiarism case is detected. But for argument's sake, we follow the reasoning of Grozea and Popescu that this ranking is counterintuitive; they then generalize from the above example and claim that $plagdet$ ranks two plagiarism detectors A and B *always counterintuitive* if $prec_A = prec_B$, $rec_A < rec_B$, $gran_A = 1$, and $gran_B > 1$.[12] Again, in plain terms, their claim is that, under $plagdet$, a plagiarism detector with perfect granularity is *always counterintuitively* ranked higher than one with higher recall and less than perfect granularity, when keeping precision fixed. To further substantiate their claim, Grozea and Popescu present at plot in which they attempt to visualize $plagdet$'s alleged area of counterintuitivity in precision-recall space: for this, detector A is fixed at perfect granularity $gran_A = 1$, varying precision and recall. Detector B is fixed at perfect recall $rec_B = 1$, a granularity $gran_B = 2$, varying precision in accordance with the above condition $prec_A = prec_B$. Observe that this setting fulfills the above conditions for counterintuitivity. Next, both detectors' $plagdet$ scores for pairs of precision and recall are computed, and their differences $plagdet_A - plagdet_B$ are visualized at the respective points in precision-recall space by means of color. The left contour plot of Figure 4 illustrates this procedure.[13] Grozea and Popescu then state that the area above the 0-contour is $plagdet$'s area of counterintuitivity (i.e., the $plagdet$ of detector A is better than that of detector B). Moreover, it is claimed that, because of the area above the 0-contour filling more than 50% of the precision-recall space, the alleged problem is severe, presuming the probability of reaching a given point in precision-recall space is evenly distributed.

In what follows, we pick up the analysis where Grozea and Popescu left off and show that the problem is in fact not that severe. To begin with, observe that the entire left half of the precision-recall space, where precision is below 0.5, is not interesting in practice. This area corresponds to plagiarism detectors for which more than half of the text reported as plagiarism actually is not plagiarized. To be considered useful, a detector should achieve precision at least above 0.5. The ranking of poor performing detectors below that is unimportant. Further note that, when the recall of detector A approaches the perfect recall of detector B, their ranking becomes less and less counterintuitive until, at perhaps $rec_A = 0.8$, humans would start preferring detector A over B for A's perfect granularity and despite its less than perfect recall. Say, the higher the recall, the more important other performance measures become. These considerations significantly reduce the size of the critical area in the precision-recall space, as shown left in Figure 4. However, the critical area can be reduced even further when adjusting detector B to be more realistic. When choosing $rec_B = 0.7$ which is the best recall achieved this year, and $gran_B = 1.1$, which is this year's average granularity (exclud-

[12] Grozea and Popescu add another condition $plagdet_A > plagdet_B$, which follows directly from the above conditions and can hence be omitted.

[13] The plot of Grozea and Popescu shows only the contour labeled 0, say, the line at which the ranking of A and B under $plagdet$ switches.

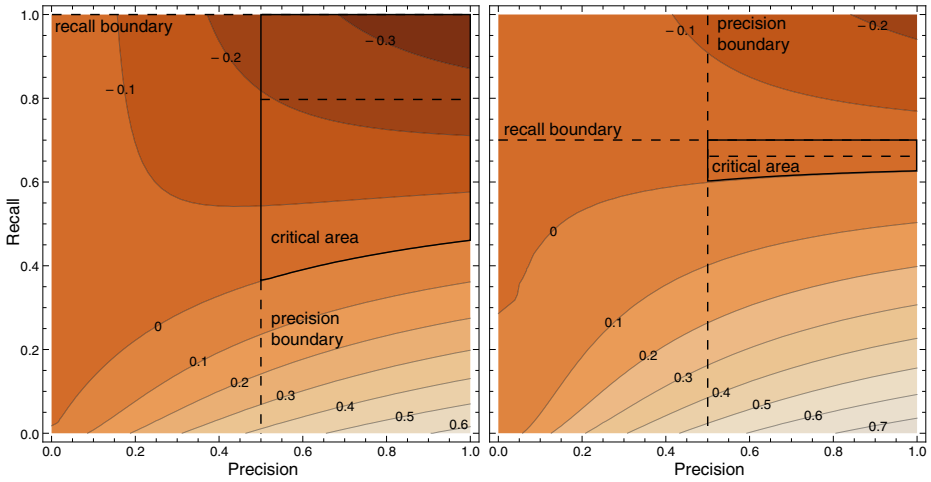

Figure 4. Contour plots of the differences of the $plagdet$ scores of two plagiarism detectors A and B in the precision-recall space. The left plot shows the situation when setting $gran_A = 1$, $rec_B = 1$, $gran_B = 2$, and $prec_A = prec_B$. The right plot shows the situation when setting $gran_A = 1$, $rec_B = 0.7$, $gran_B = 1.1$, and $prec_A = prec_B$. The line markings delineate a critical area in which detectors A and B might be ranked counterintuitively under $plagdet$.

ing the detector of Jayapal [14]), the resulting $plagdet$ differences of detectors A and B are less pronounced (see right plot in Figure 4). Still, a precision below 0.5 remains uninteresting, and a recall $rec_A > 0.7$ violates the above condition that $rec_A < rec_B$. Thus, the critical area becomes much smaller than before, and even more so when taking into account that, when rec_A approaches rec_B, Grozea and Popescu's claim that rankings within the critical area are always counterintuitive turns out to be false. Moreover, the actual $plagdet$ differences between detectors A and B are much smaller than 0.1, so that rankings of more than one detector may only be affected locally, but not globally.

Altogether, we conclude that the problem of possible counterintuitive rankings of plagiarism detectors under $plagdet$ is not as severe as previously thought. Hence, we believe that introducing a new hyper-parameter to counter this issue—as is proposed by Grozea and Popescu—is not of primary concern, and would only add confusion. In general, however, there is still a lot of room to improve existing performance measures and to invent new ones. In the future, we will again take a closer look in this direction.

3.2 Survey of Detection Approaches

Eleven of the 15 participants submitted softwares for the detailed comparison task, ten of whom also submitted at notebook describing their approach. An analysis of these notebooks reveals a number of building blocks that were commonly used to build detailed comparison algorithms: (1) seeding, (2) match merging, and (3) extraction filtering. In what follows, we describe them in detail.

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

A number of seed heuristics have been applied by this year's participants: Kong et al. [17] use sentences from the suspicious and source document whose surrounding passages have a similarity above some threshold and whose sentence-overlap is above another threshold. Suchomel et al. [32] use sorted word 5-grams and unsorted stop word 8-grams (the latter having been introduced in [30]). Grozea and Popescu [11] and Obereuter et al. [19] use char 16-grams and char 18-grams. Rodríguez Torrejón and Martín Ramos [28] use sorted word 3-grams and sorted word 1-skip-3-grams. Palkovskii and Belov [20] use word 3-grams, Küppers and Conrad [18] use non-overlapping 250 char chunks whose word-based similarity under Dice's coefficient is above some threshold, Sánchez-Vega et al. [29] use single words, and Jayapal [14] uses char n-grams where n ranges from 3 to 15. Before computing seeds, some participants choose to collapse whitespace, reduce cases, remove stop words, and stem the remaining words, if applicable to their respective seed heuristics. However, the idea of synonym normalization used in previous years appears to have been forgotten or discarded.

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

Most of the participants' match merging heuristics are rule-based, merging seeds into aligned passages if they are adjacent in both suspicious and source document and the size of the gap between them is below some threshold. The exact rule depends on the seeds used, and instead of using just one rule, many participants develop sets of constraints that have to be fulfilled by aligned passages in order to be reported as plagiarism detections. Since the rules are usually highly involved with their respective setup, we exemplify only one rule set here in order to give an idea of what they may look like: Suchomel et al. [32] employ a 2-step merge heuristic, where in the first step, adjacent seed matches that are no more than 4000 chars apart are merged. The resulting passages from the first step are then merged again, considering pairs of adjacent passages in turn, and checking if the gap between them contains at least four seeds so that there is at least one seed per 10 000 chars of gap length between them. To be merged, adjacent passages further have to fulfill the constraints that their gap is smaller than 30 000 chars, that their combined size is bigger than twice the gap size, and that the ratio of seeds per chars of the adjacent passages does not drop by a factor of more than three in the potentially merged passage. The only participants who go beyond rule-based merging are Grozea and Popescu [11], who combine rules with randomization, and Palkovskii and Belov [20], who employ clustering algorithms for unsupervised merging.

Passage Filtering. Given a set of aligned passages, a passage filter removes all aligned passages that do not meet certain criteria. Rationale for this is mainly to deal with overlapping passages and to discard extremely short passages. Kong et al. [17] discard passages whose word overlap under a modified Jaccard coefficient is below a threshold. Suchomel et al. [32] discard overlapping passages that are shorter than 300 chars, and keep only the passages longer than 300 chars. Oberreuter et al. [19] discard passages shorter than 120 chars and Palkovskii and Belov [20] discard passages shorter than 190 chars. Gillam et al. [7] discard passages shorter than 50 words that have less than 0.75 cosine similarity under a vector space model. The other participants do not apply passage filtering.

Remarks. Since seven of this year's participants have taken part in previous competitions as well, many of them have simply reused their earlier solutions, some repeatedly. Others have simply used existing algorithms like greedy string tiling and BLAST out of the box. While there is no problem with doing so, this indicates that participants are maybe at a loss about how to further improve their approaches or devise new ones. That said, the winning approach of Kong et al. [17] is entirely new to the competition, but unfortunately no details are disclosed about it because of a patent pending.

The detailed comparison of documents for plagiarism detection is closely related to sequence alignment in bioinformatics, of which the terminology used above is borrowed. Oberreuter et al. [19] are the first to explore this connection by applying one of the algorithms from the well-known BLAST family. Moreover, a number of new ideas could be observed:

- Sánchez-Vega et al. [29] apply scored seeding and merge seed matches based on their scores instead of just their proximity. This idea is also relates to sequence alignment algorithms.
- Palkovskii and Belov [20] and Jayapal [14] try to adjust their approaches differently based on the situation at hand (i.e., based on how strong a plagiarism case is obfuscated). Although the two approaches, in their current state, are not that successful, this idea points into a promising new direction for tailoring detailed comparison algorithms to certain situations instead of developing a one-fits-all approach.
- Suchomel et al. [32] and Palkovskii and Belov [20] are the first to employ more than one seed heuristic at the same time. This shows that combining multiple sources of information may further help to devise better detailed comparison algorithms.
- Suchomel et al. [32] and Palkovskii and Belov [20] merge seed matches not in one but two steps. This iterative merging of seeds allows for using different merge heuristics at different levels of abstraction, thus reducing the risk of making errors. Again, a one-fits-all approach is probably more difficult to develop and maintain.

Finally, regarding the detection of cross-language plagiarism, most participants simply resorted to using the translated version of a document obtained from Google Translate which was provided as an additional parameter in the test phase. Some disregarded non-English cases altogether, while only Rodríguez Torrejón and Martín Ramos [28] continued to develop their own dictionary-based solution.

Table 3. Implementation details of the detailed comparison submissions, sorted by runtime.

Team	Submission Size [MB]	Operating System	Programming Language	Average Runtime [sec/comparison]
Rodríguez Torrejón	1.80	Linux	sh, C/C++	0.19
Sánchez-Vega	0.04	Linux	C++	2.48
Oberreuter	0.19	Linux	Java	2.58
Palkovskii	68.20	Windows	C#	4.51
Grozea	1.90	Linux	Perl, Octave	4.82
Suchomel	0.02	Linux	Perl	5.36
Kong	2.60	Linux	Java	5.91
Jayapal	37.20	Linux	Java	8.43
Gillam	0.48	Linux	Python 2.7	9.40
Küppers	42.90	Linux	Java	27.64
Ghosh	554.50	Linux	sh, Java	–

3.3 Evaluation Results

The softwares submitted to the detailed comparison subtask vary widely with respect to their sizes, employed programming languages, and runtime performances (cf. Table 3 for details). Regarding runtime, the submission of Rodríguez Torrejón and Martín Ramos [28] is outstanding: it achieves 0.19 seconds per comparison, which is an order of magnitude faster than all other submissions. The two fastest approaches are both implemented in C/C++. Note that we always tried to run a submission on a Linux system and resorted to Windows only if necessary.

Table 4 shows the detection performances of the ten evaluated detailed comparison approaches. Besides the overall performance on the complete evaluation corpus (first value in brackets), also the performance on each sub-corpus is given (remaining values in brackets). Note that for the sub-corpus without plagiarism, no performance values can be stated due to the lack of true positives. However, false positive detections for this sub-corpus influenced the overall performance of course. The winner of this year's detailed comparison task is the approach by Kong et al. [17], which achieves the highest $plagdet$ score on the complete corpus, the real cases, the simulated paraphrase cases, as well as on the cross language cases. The best performing approach on the artificial paraphrase cases (both high and low) was submitted by Oberreuter et al. [19], whose overall performance suffered from a bad performance on the cross-language cases. Suchomel et al. [32], who submitted the second best approach, performed best on the plagiarism cases without any obfuscation. Regarding real plagiarism cases, the ranking from the second place onwards would change; here, Rodríguez Torrejón and Martín Ramos [28] achieve the best recall. Most approaches perform better on real plagiarism cases compared to their overall performance. However, these results must be taken with a grain of salt, since the statistical mass of 33 real cases is too small to claim good performance in detecting real plagiarism. Nevertheless, the inclusion of real cases adds more realism to our evaluation, and we strive to scale up their number in the future.

Table 4. Performances on the detailed comparison subtask. For each performance metric, the achieved score is stated in the first entry inside the brackets for each team. The remaining entries in each bracket refer to the sub-corpora Real Cases, Simulated Paraphrases, Translations, Artificial Paraphrases High, Artificial Paraphrases Low, and No Obfuscation. The best score in each column is highlighted.

Rank / Team	**PlagDet**	**Precision**	**Recall**	**Granularity**
	[Overall ,	Real Cases, Simulated, Translation,	Artificial High, Artificial Low,	No Obfuscation]
1 Kong	[**.738** , **.756**, **.758**, **.771**, .396, .822, .898]	[.824 , .964, .900, .820, .750, .928, .834]	[**.678** , .621, **.657**, **.727**, **.277**, .777, .973]	[1.01 , 1.00, 1.00, 1.00, 1.02, 1.04, 1.00]
2 Suchomel	[.682 , .624, .611, .748, .153, .801, **.938**]	[.893 , .869, .915, .956, .793, .948, .885]	[.552 , .487, .458, .614, .085, .694, .999]	[1.00 , 1.00, 1.00, 1.00, 1.00, 1.00, 1.00]
3 Grozea	[.678 , .636, .668, .713, .380, .753, .848]	[.774 , .667, .888, .709, .818, .879, .753]	[.635 , .607, .536, .716, .272, .778, .969]	[1.03 , 1.00, 1.00, 1.00, 1.10, 1.13, 1.00]
4 Oberreuter	[.673 , .713, .717, .006, **.407**, **.844**, .923]	[.867 , .857, .907, **1.00**, **.873**, **.955**, .857]	[.555 , .611, .593, .003, .267, **.781**, **.999**]	[1.00 , 1.00, 1.00, 1.00, 1.00, 1.02, 1.00]
5 Rodríguez Torrejón	[.625 , .740, .673, .257, .127, .713, .932]	[.834 , .830, .811, .985, .764, .896, .898]	[.500 , **.668**, .577, .148, .069, .593, .968]	[1.00 , 1.00, 1.00, 1.00, 1.00, 1.00, 1.00]
6 Palkovskii	[.538 , .710, .619, .000, .318, .769, .813]	[.574 , .810, .725, .000, .688, .859, .707]	[.523 , .631, .541, .000, .211, .751, .998]	[1.02 , 1.00, 1.00, 1.00, 1.02, 1.05, 1.02]
7 Küppers	[.349 , .521, .299, .001, ,026, .266, .809]	[.776 , .776, **.963**, .679, .276, .930, .817]	[.282 , .392, .244, .000, .013, .192, .934]	[1.26 , 1.00, 1.46, 1.00, 1.00, 1.29, 1.10]
8 Sánchez-Vega	[.309 , .509, .264, .397, .042, .253, .603]	[.537 , .564, .843, .839, .186, .576, .592]	[.349 , .464, .245, .433, .030, .282, .847]	[1.57 , 1.00, 1.71, 1.71, 1.36, 1.81, 1.22]
9 Gillam	[.308 , .035, .124, .000, .009, .047, .929]	[**.898** , **1.00**, .915, .000, .739, .917, .926]	[.190 , .018, .066, .000, .007, .027, .935]	[1.02 , 1.00, 1.00, 1.00, 1.88, 1.20, 1.00]
10 Jayapal	[.045 , .233, .044, .000, .013, .050, .102]	[.622 , .783, .485, .000, .240, .537, **.988**]	[.075 , .263, .049, .000, .011, .062, .245]	[6.93 , 2.23, 3.10, 1.00, 2.07, 3.58, 14.3]

4 Summary

To sum up the fourth international competition on plagiarism detection at PAN'12, we have introduced a number of novelties: a new evaluation framework that approaches plagiarism detection step-wise instead of as a whole, and that allows for software submissions instead of just result submissions. We are the first to consider the heretofore neglected, yet important scenario of plagiarism detection from the web at a representative scale. Furthermore, we introduce new tools to evaluate plagiarism detectors, namely the ChatNoir search engine, and the TIRA experimentation platform. Both allow for assessing a plagiarism detector's performance based on new measures, such as runtime and web retrieval performance. We have constructed a new, large plagiarism corpus based on crowdsourcing that consists of entirely manually written plagiarism cases. Our corpus construction pipeline emulates the entire process of plagiarizing text from the web in a controlled environment. The data collected allows for a first time glimpse over the shoulders of plagiarists. Finally, we have used real plagiarism cases to evaluate detection performance, which was made possible by our new evaluation tools that do not require test data to be handed out to participants. Otherwise, the use of real plagiarism for evaluation would cause legal and ethical problems.

We have demonstrated that a step-wise evaluation of plagiarism detectors is the way forward, whereas the two plagiarism detection subtasks candidate retrieval and detailed comparison pose new challenges for evaluation. In the coming competitions, we plan on improving our evaluation framework in order to reach a new stable point at which evaluations within the framework can be run smoothly out of the box. In particular, we will encourage software submissions not only for detailed comparison but also for candidate retrieval, again using the TIRA experimentation platform to facilitate this goal. Our vision is to implement a fully automatic, web-based plagiarism detection evaluator, available to all researchers in this field.

Acknowledgements

We thank the participants of PAN for their dedicated work, and for being patient with our ever changing idea of how plagiarism detectors should be evaluated. This work was partly funded by the EC WIQ-EI project (project no. 269180) within the FP7 People Program, by the MICINN Text-Enterprise (TIN2009-13391-C04-03) research project, and by the ERCIM "Alain Bensoussan" Fellowship Programme (funded from the European Union Seventh Framework Programme FP7/2007-2013 under grant agreement number 246016).

Bibliography

[1] Jeff Barr and Luis Felipe Cabrera. AI gets a Brain. *Queue*, 4(4):24–29, May 2006.

[2] Paul Clough and Mark Stevenson. Developing a Corpus of Plagiarised Short Answers. *Lang. Resour. Eval.*, 45:5–24, March 2011. ISSN 1574-020X. doi: 10.1007/s10579-009-9112-1.

[3] Paul Clough, Robert Gaizauskas, Scott S. L. Piao, and Yorick Wilks. METER: MEasuring TExt Reuse. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 152–159, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073110.

[4] Gordon V. Cormack, Mark D. Smucker, and Charles L. A. Clarke. Efficient and effective spam filtering and re-ranking for large web datasets. *Information Retrieval*, 14(5):441–465, 2011.

[5] Tamer Elsayed, Jimmy J. Lin, and Donald Metzler. When close enough is good enough: approximate positional indexes for efficient ranked retrieval. In *Proceedings of the 20th ACM Conference on Information and Knowledge Management, CIKM 2011, Glasgow, United Kingdom, October 24-28, 2011*, pages 1993–1996, 2011.

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

[7] Lee Gillam, Neil Newbold, and Neil Cooke. Educated Guesses and Equality Judgements: Using Search Engines and Pairwise Match for External Plagiarism Detection—Notebook for PAN at CLEF 2012. In Forner et al. [6]. ISBN 978-88-904810-3-1. URL http://www.clef-initiative.eu/publication/working-notes.

[8] Tim Gollub, Steven Burrows, and Benno Stein. First Experiences with TIRA for Reproducible Evaluation in Information Retrieval. In Andrew Trotman, Charles L. A. Clarke, Iadh Ounis, J. Shane Culpepper, Marc-Allen Cartright, and Shlomo Geva, editors, *SIGIR 12 Workshop on Open Source Information Retrieval (OSIR 12)*, pages 52–55, August 2012.

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

[10] Tim Gollub, Benno Stein, Steven Burrows, and Dennis Hoppe. TIRA: Configuring, Executing, and Disseminating Information Retrieval Experiments. In A Min Tjoa, Stephen Liddle, Klaus-Dieter Schewe, and Xiaofang Zhou, editors, *9th International Workshop on Text-based Information Retrieval (TIR 12) at DEXA (to appear)*, September 2012.

[11] Cristian Grozea and Marius Popescu. Encoplot - Tuned for High Recall (also proposing a new plagiarism detection score). In Forner et al. [6]. ISBN 978-88-904810-3-1. URL http://www.clef-initiative.eu/publication/working-notes.

[12] Matthias Hagen and Benno Stein. Candidate Document Retrieval for Web-Scale Text Reuse Detection. In *18th International Symposium on String Processing and Information Retrieval (SPIRE 11)*, volume 7024 of *Lecture Notes in*

Computer Science, pages 356–367. Springer, 2011. doi: http://dx.doi.org/10.1007/978-3-642-24583-1_35.

[13] Djoerd Hiemstra and Claudia Hauff. MIREX: MapReduce information retrieval experiments. Technical Report TR-CTIT-10-15, University of Twente, 2010.

[14] Arun kumar Jayapal. Similarity Overlap Metric and Greedy String Tiling at PAN 2012: Plagiarism Detection—Notebook for PAN at CLEF 2012. In Forner et al. [6]. ISBN 978-88-904810-3-1. URL http://www.clef-initiative.eu/publication/working-notes.

[15] Patrick Juola. An Overview of the Traditional Authorship Attribution Subtask. In Forner et al. [6]. ISBN 978-88-904810-3-1. URL http://www.clef-initiative.eu/publication/working-notes.

[16] Philipp Koehn. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Conference Proceedings: the tenth Machine Translation Summit*, pages 79–86, Phuket, Thailand, 2005. AAMT, AAMT. URL http://mt-archive.info/MTS-2005-Koehn.pdf.

[17] Leilei Kong, Haoliang Qi, Shuai Wang, Cuixia Du, Suhong Wang, and Yong Han. Approaches for Candidate Document Retrieval and Detailed Comparison of Plagiarism Detection—Notebook for PAN at CLEF 2012. In Forner et al. [6]. ISBN 978-88-904810-3-1. URL http://www.clef-initiative.eu/publication/working-notes.

[18] Robin Küppers and Stefan Conrad. A Set-Based Approach to Plagiarism Detection—Notebook for PAN at CLEF 2012. In Forner et al. [6]. ISBN 978-88-904810-3-1. URL http://www.clef-initiative.eu/publication/working-notes.

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

[20] Yurii Palkovskii and Alexei Belov. Applying Specific Clusterization and Fingerprint Density Distribution with Genetic Algorithm Overall Tuning in External Plagiarism Detection—Notebook for PAN at CLEF 2012. In Forner et al. [6]. ISBN 978-88-904810-3-1. URL http://www.clef-initiative.eu/publication/working-notes.

[21] Martin Potthast. *Technologies for Reusing Text from the Web*. Dissertation, Bauhaus-Universität Weimar, December 2011.

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

[23] Martin Potthast, Alberto Barrón-Cedeño, Andreas Eiselt, Benno Stein, and Paolo Rosso. Overview of the 2nd International Competition on Plagiarism Detection. In Martin Braschler, Donna Harman, and Emanuele Pianta, editors, *Notebook Papers of CLEF 10 Labs and Workshops*, September 2010. ISBN

978-88-904810-2-4. URL
http://www.clef-initiative.eu/publication/working-notes.

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

[25] Martin Potthast, Andreas Eiselt, Alberto Barrón-Cedeño, Benno Stein, and Paolo Rosso. Overview of the 3rd International Competition on Plagiarism Detection. In Vivien Petras, Pamela Forner, and Paul D. Clough, editors, *Notebook Papers of CLEF 11 Labs and Workshops*, September 2011. ISBN 978-88-904810-1-7. URL http://www.clef-initiative.eu/publication/working-notes.

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

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

[28] Diego A. Rodríguez Torrejón and José Manuel Martín Ramos. Detailed Comparison Module In CoReMo 1.9 Plagiarism Detector—Notebook for PAN at CLEF 2012. In Forner et al. [6]. ISBN 978-88-904810-3-1. URL http://www.clef-initiative.eu/publication/working-notes.

[29] Fernando Sánchez-Vega, Manuel Montes y Gómez, and Luis Villaseñor-Pineda. Optimized Fuzzy Text Alignment for Plagiarism Detection—Notebook for PAN at CLEF 2012. In Forner et al. [6]. ISBN 978-88-904810-3-1. URL http://www.clef-initiative.eu/publication/working-notes.

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

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

[32] Šimon Suchomel, Jan Kasprzak, and Michal Brandejs. Three Way Search Engine Queries with Multi-feature Document Comparison for Plagiarism Detection—Notebook for PAN at CLEF 2012. In Forner et al. [6]. ISBN 978-88-904810-3-1. URL http://www.clef-initiative.eu/publication/working-notes.