Source Retrieval for Web-Scale Text Reuse Detection

Matthias Hagen Martin Potthast Payam Adineh Ehsan Fatehifar Benno Stein
Bauhaus-Universität Weimar
99421 Weimar, Germany
⟨firstname⟩.⟨lastname⟩@uni-weimar.de

ABSTRACT

The first step of text reuse detection addresses the source retrieval problem: given a suspicious document, a set of candidate sources from which text might have been reused have to be retrieved by querying a search engine. Afterwards, in a second step, the retrieved candidates run through a text alignment with the suspicious document in order to identify reused passages. Obviously, any true source of text reuse that is not retrieved during the source retrieval step reduces the overall recall of a reuse detector. Hence, source retrieval is a *recall-oriented task*, a fact ignored even by experts: Only 3 of 20 teams participating in a respective task at PAN 2012–2016 managed to find more than half of the sources, the best one achieving a recall of only 0.59. We propose a new approach that reaches a recall of 0.89—a performance gain of 51%.

1 INTRODUCTION

Identifying text reuse is a key component in plagiarism detection, copyright protection, information flow analysis, and related tasks. Usually, text reuse detection is a three-step process [27]:

(1) *Source retrieval*, where candidate source documents for a given suspicious document are retrieved from a large text collection.
(2) *Text alignment*, where text passages similar to passages in the suspicious document are extracted from the candidate sources.
(3) *Knowledge-based post-processing*, where the extracted pairs of passages (from a candidate document and the suspicious document) are analyzed with regard to citation information, and visualized for convenient inspection.

If the universe of documents from which reuse is to be detected is small (say, less than 100,000 documents), an in-depth comparison of every document against the suspicious document is feasible and takes only minutes using modern text alignment approaches, so that Step (1) can be skipped. If the universe is large (say, web-scale), source retrieval is necessary to narrow down the set of candidate documents that undergo text alignment to a feasible size.

Step (1) forms the setting of a corresponding shared task we have organized at PAN, with special emphasis on web retrieval. The participants had to use the API of a standard web search engine to submit queries from whose results a "promising" set of candidate sources for a given suspicious document could be requested as downloads. This setting introduces an interesting optimization problem: a successful approach must achieve a high true-source recall within the downloaded candidates, since the analysis steps that follow can exploit only the candidate set, while it should also minimize the amount of submitted queries, since using the APIs of search engines incurs costs per query. The task is motivated by a real-world scenario: a detection service has to pay for search engine API requests, such that a lower number of queries means reducing operational costs as well as gaining a competitive advantage by reducing the analysis prices for end users.

The PAN source retrieval task gathered solutions from 20 teams in the years 2012–2016. Apparently, most participants focused on minimizing the effort in form of queries and downloads, rather than on maximizing the recall. However, according to our understanding of the realities, recall still comes first, in order to allow for a high detection quality, while query and download effort is secondary—an interpretation not only propagated in our shared task overviews [7, 19, 21, 22] but also in the similar TREC Total Recall tracks [5]. Downloading many results and creating a candidate set of even tens of thousands of documents is no real bottleneck for text reuse detection as text alignment at that scale is handled very efficiently by modern approaches [19]. Even a higher query effort can be tolerated if it achieves a substantially higher recall. For the first time, our approach implements a recall-oriented source retrieval strategy, achieving a very strong recall of 0.89 that significantly outperforms the previously best recall of 0.59. Based on this result, we propose effort reduction strategies that render our approach comparable to others in terms of effort at a recall of still 0.76.

2 BACKGROUND AND RELATED WORK

After introducing the source retrieval evaluation framework, we survey the strategies employed by the PAN shared task participants.

2.1 Source Retrieval Evaluation Framework

The source retrieval evaluation framework has four components: (1) a static web search environment, (2) an evaluation corpus, (3) an evaluation-as-a-service platform, and (4) performance measures.

(1) The search environment consists of the English portion of the ClueWeb09 web collection[1] and two search engines that index its 500 million pages, namely Indri[1] and ChatNoir [20]. (2) The evaluation corpus is based on the Webis Text Reuse Corpus 2012 (Webis-TRC-12) [23] which comprises essays whose authors reused text passages from the English portion of the ClueWeb09; 98 of these essays form the training set, and another 99 essays form the test set of the source retrieval framework. (3) TIRA [4] has been employed as an evaluation-as-a-service platform at PAN's source retrieval task in order to collect the participants' softwares. When deployed on TIRA, an approach can query the provided search

[1] http://lemurproject.org/clueweb09

CIKM'17 , Singapore, Singapore

978-1-4503-4918-5/17/11…$15.00
DOI: 10.1145/3132847.3133097

engine APIs and request downloads from the ClueWeb09 while its activities are logged. For each requested download, a source oracle gives feedback whether it is a true source or not, eliminating the need of implementing text alignment. (4) The framework also comprises evaluation scripts that return macro-averaged recall and effort scores (total number of queries and downloads per suspicious document, as well as numbers of queries and downloads up to a first true detection) following the evaluation procedure of the PAN source retrieval task [7, 19, 21, 22], and comparable to the TREC Total Recall track's evaluation [5].

2.2 Survey of Source Retrieval Approaches

Overall, 20 teams participated in the PAN source retrieval task. All followed a similar algorithmic scheme: chunking, keyphrase extraction, query formulation, and query/download scheduling.

Chunking. A suspicious document is divided into—typically non-overlapping—passages, called "chunks." Each chunk in turn is then processed individually. The rationale for chunking a suspicious document is to evenly distribute "attention" over the document. The chunking strategies employed by the participants range from no chunking (the whole document is treated as a single chunk) [1, 14, 17, 28–31, 35], sections identified by intrinsic plagiarism detection techniques [18], paragraphs as chunks [15–17, 26, 29, 31], 50-line chunks [1, 2], 500-word chunks [25], 200-word chunks [24], 100-word chunks [24, 32], 25-sentence chunks [3], 5-sentence chunks [12, 17, 33, 34], 4-sentence chunks [6, 11], to individual sentences and headings as chunks [10, 13].

Keyphrase Extraction. Given a chunk, keyphrases are extracted to formulate queries with them. The idea is to select only those phrases (or words) that maximize the chance of retrieving "promising" candidate source documents. Note that the fewer phrases are extracted and the longer the chunks are, the more elaborate an extraction policy must be; otherwise, recall will be harmed. Some participants use single words while others extract phrases. Most of the participants preprocessed the suspicious document by removing stop words before applying a keyphrase extraction algorithm. Ideas are to use the first words of a chunk [18], to use POS-taggers to extract words with specific POS-tags like nouns or adjectives [10, 11, 13, 33–35], to select rare words [3, 6, 16, 32], to detect named entities [1, 2, 17], to use keyphrase extraction techniques from the NLP community [1, 2, 6], to select complete stopped headings or stopped sentences with rich vocabulary [28, 30, 31], or to simply use words with high tf [24] or $tf \cdot idf$ values [1, 2, 12, 14, 15, 25, 26, 28–31].

Query Formulation. Interestingly, the actual query formulation is rather simplistic for all participating approaches. Typically, the top-k terms from some ranking of extracted keyphrases for a chunk form the first query, then the next k terms, etc. (usually $k \leq 10$). This way, mostly non-overlapping queries are generated for the individual chunks. However, as the goal is to find sources containing similar (often short) text passages, simply merging terms to queries that occur in the same chunk seems sufficient, so that more sophisticated schemes to formulate queries appear unnecessary. The differences between the strategies employed are more or less in the number of queries and the level for which they are computed. For instance, some participants formulate one query per sentence of a chunk [10, 13, 25], one to many queries per chunk [26, 29], or even some more general queries at the document level [17, 29].

Query and Download Scheduling. Given a set of queries, the scheduling manages their submission to the search engine and decides which results to actually download. Rationale for this is to be able to dynamically adjust the process based on previous results, which could range from not downloading some result or dropping a whole pre-computed query to reformulating queries or even formulating new ones based on the relevance feedback obtained from the search results. Regarding the querying strategies applied by the PAN participants, most simply submit all the queries in the order of the original chunks for which they were computed, while some drop queries similar to previous queries or already downloaded results [24–26, 29, 35], and some reorder the queries starting with more general ones for an early retrieval of at least some source [28–31]. Regarding the downloading strategies, some approaches simply download all of the up to 100 available results [10, 13, 15], some are very "picky" and only download the top-k results for some $k \ll 100$ [2, 3, 11, 12, 16, 17, 24, 25, 32, 33, 35], some only download results when a provided long snippet is sufficiently similar to the suspicious document [1, 6, 10, 14, 17, 26, 29, 31, 34], or when some trained classifier decides to download [34].

3 WEBIS SOURCE RETRIEVAL 2017 (WSR17)

Following the outlined scheme, we propose the Webis Source Retrieval 2017 approach (WSR17), developing two variants: one maximizing recall paying less attention to effort in terms of queries and downloads (high-recall WSR17), and based on the former, one that trades as little recall as possible to reduce effort (trade-off WSR17). For reproducibility sake, the code for our approach is publicly available,[2] as well as deployed to TIRA.

Chunking. We use Lucene to detect a document's paragraph structure[3] and apply post-processing strategies that were tested in pilot experiments on the training set. For the high-recall approach, the post-processing splits paragraphs with more than 150 words into non-overlapping 150-word chunks (the resulting "last" chunk of a paragraph possibly being shorter). This somewhat evenly distributes attention also for longer paragraphs.

The post-processing for the trade-off approach tries to avoid short paragraphs (e.g., individual bullet points) that can cause many queries (e.g., for a longer list of bullet points). If a paragraph has less than 50 words, it is joined with the respective next paragraph(s) before the splitting of chunks with more than 150 words is started.

Keyphrase Extraction. Both variants of our approach use the same keyphrase extraction at chunk level but differ in their usage of document-level keyphrases. As for the chunk-level keyphrases, we follow the majority of the PAN participants who used $tf \cdot idf$ scoring for keyphrase extraction: Employing NLTK's stopword removal,[4] a chunk's top-10 words according to $tf \cdot idf$ scores are selected. We compute idf based on the English portion of the ClueWeb09, whereas the PAN participants relied on way smaller corpora.

Some PAN participants extracted document-level keyphrases which yielded promising results in our own pilot experiments on the training set. For our high-recall variant, we extract the top-20 unigrams, the top-10 bigrams, the top-5 trigrams, the top-5

[2] https://github.com/webis-de/ [3] https://lucene.apache.org/ [4] www.nltk.org/

Table 1: Experimental results sorted by macro-averaged recall. Coverage measures the fraction of test documents for which at least one of its sources was retrieved. Effort is given macro-averaged per suspicious document and until first detection.

Participant	Recall (macro-averaged)	Coverage	Effort (per document)		Effort (1st detection)	
			Queries	Candidates	Queries	Candidates
ⓐ High-recall WSR17	**0.89**	**1.00**	553.1	41823.6	18.3	182.0
ⓑ Trade-off WSR17	**0.76**	**1.00**	180.2	2588.2	15.3	91.5
ⓒ Kong13	0.59	**1.00**	47.9	5185.3	**2.5**	210.2
ⓓ Maluleka16	0.53	0.94	138.6	18.7	20.9	2.2
ⓔ Prakash14	0.51	0.93	60.0	38.8	8.1	3.8
ⓕ Kong14	0.48	0.94	83.5	207.1	85.7	24.9
ⓖ Williams14	0.48	0.96	117.1	14.4	18.8	2.3
ⓗ Williams13	0.47	0.93	117.1	12.4	23.3	2.2
ⓘ Zubarev14	0.45	0.97	37.0	18.6	5.4	2.3
ⓙ Suchomel15	0.43	0.96	42.4	359.3	3.3	39.8
ⓚ Kong15	0.42	0.97	195.1	38.3	197.5	3.5
ⓛ Rafiei15	0.41	0.99	43.5	183.3	5.6	24.9
ⓜ Suchomel14	0.40	0.98	19.5	237.3	3.1	38.6
ⓝ Sanjesh15	0.39	0.92	90.3	8.5	17.5	1.6
ⓞ Elizalde14	0.39	0.93	54.5	33.2	16.4	3.9
ⓟ Elizalde13	0.37	0.96	41.6	83.9	18.0	18.2
ⓠ Lee13	0.37	0.91	48.4	10.9	6.5	2.0
ⓡ Han15	0.32	0.88	194.5	11.8	202.0	1.7
ⓢ Haggag13	0.31	0.88	41.7	**5.2**	13.9	**1.4**
ⓣ Foltynek13	0.26	0.68	166.8	72.7	180.4	4.3
ⓤ Suchomel13	0.23	0.82	17.8	283.0	3.4	64.9
ⓥ Gillam13	0.15	0.66	**15.7**	86.8	16.1	28.6

four-grams, and the top-20 five-grams at document level according to $tf \cdot idf$ scores. The different amounts of n-grams reflect their contribution to recall as per our pilot experiments. Hence, in our trade-off approach, we omit bigrams, trigrams, and four-grams to reduce overall effort in terms of queries and downloads.

Query Formulation. At chunk level, both variants formulate two queries per chunk: the top-5 keywords in a first query, the other 5 keywords in a second query. As for the document level queries, both formulate 4 unigram queries (the top-5 unigrams in a first query, etc.), and each five-gram as a single query (i.e., 20 queries). The high-recall approach also formulates five bigram queries (the top-2 bigrams in a first query, etc.) and each tri- and four-gram as a single query (i.e., another ten queries). Employing a more sophisticated combination technique [9] for document-level queries could be an interesting direction for future research.

Query and Download Scheduling. Since document-level queries ensured early detection of at least some source in our pilot experiments on the training set, we first submit document-level queries (first all unigram queries, then all bigram queries, etc.) and then the chunk-level queries in the order of their chunk's appearance in the suspicious document (i.e., queries from the first chunk, then from the second chunk, etc.). As search engine, we use ChatNoir and submit batches of four queries. Queries that are identical to previous queries are not submitted again. Our trade-off approach further reduces the number of queries by not submitting queries with less than two nouns—POS-tagging done with NLTK—since queries with more nouns were more successful on the training set.

The results of every query are analyzed before submitting the next query (batch). The high-recall approach simply downloads the top-100 results (not downloading results that were already retrieved before) and then proceeds to the next query. Our trade-off approach only inspects the top-25 results since hardly any source was found below rank 25 in our pilot experiments on the training set. Additionally, the trade-off approach only downloads a document when at least one of the query's terms in stemmed form (NLTK's implementation of the Porter stemmer) appears in the stemmed ChatNoir snippet (at a default length of 500 characters).

4 EVALUATION RESULTS

To compare our system to the 20 teams who participated in the PAN shared tasks on source retrieval, we employ the recall-effort performance measures supported by the PAN framework, which is in line with the evaluation method employed in the TREC Total Recall tracks [5], where recall is the primary goal at feasible efforts.

Table 1 shows the performance scores. As can be seen, our high-recall WSR17 substantially outperforms all previous systems with respect to recall—a performance gain of about 51% over Kong13. Interestingly, our high-recall WSR17 even improves upon the ensemble of the previous methods for which a recall of 0.85 was reported [7]. Also our trade-off WSR17 improves upon the recall-wise best systems at a somewhat comparable effort (fewer downloads than Kong13, similarly many queries as Maluleka16). Given the high throughput of thousands of documents per minute achieved by modern text alignment approaches [19], even the candidate set size of our high-recall approach is manageable in a practical text reuse detector. Nevertheless, the approach of Maluleka [17] does achieve a non-trivial recall with a remarkably small candidate set.

Figure 1 shows the growth of recall dependent on the size of the candidate set (i.e., effort), considering the candidate set after each addition of a new source candidate. All plots except one have a clear s-shape and three groups can be distinguished (say, "early recall with low effort", "mid-effort", and "high recall").

An interesting line of future research is to optimize our approaches' overall effort and the effort until first detection. Early detection of some source could result in immediate feedback from

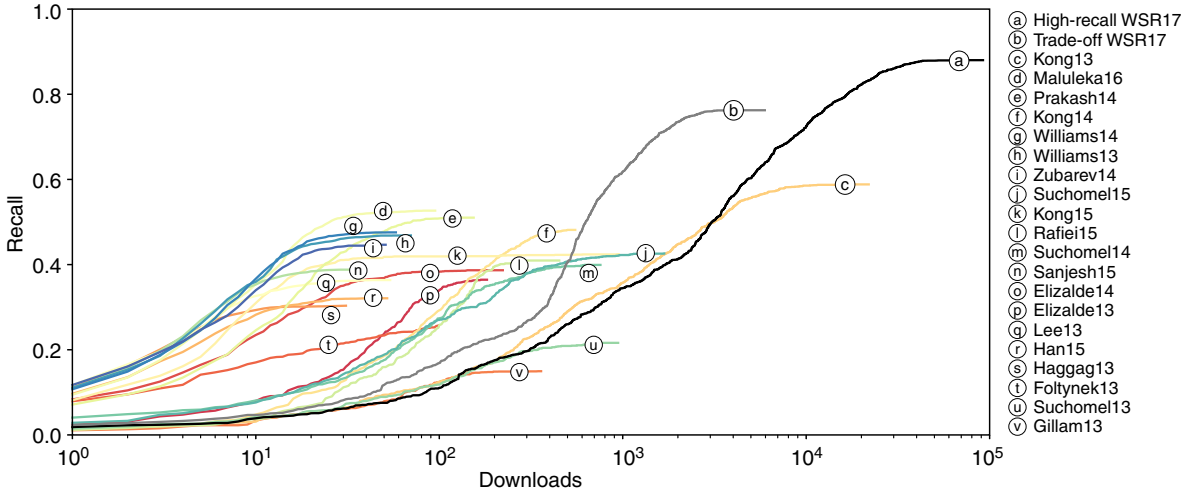

Figure 1: Recall over downloads of the 20 source retrieval systems submitted to PAN as well as our two approaches high-recall WSR17 and trade-off WSR17. Each plot shows the increase of recall after each new addition to the candidate set.

a text reuse detector to its user, indicating that a deeper analysis with more effort (and higher costs) may be worthwhile. In such a scenario, one might even start with an "early recall" system until it detects its first source, and then switch to our high-recall or trade-off WSR17 to maximize recall.

5 CONCLUSION AND OUTLOOK

Our approach sets a new standard for source retrieval recall: a 51% improvement over the previously best approach. An important direction for future research is to further reduce the effort *without sacrificing* too much recall. In particular, similarities of snippets and the suspicious document could help reducing the number of downloads while adapted query quality prediction may be of use to predict whether a query formulated from a given chunk's keyphrases will be successful or not. Also an in-depth analysis of the not-retrieved sources and the essay authors' respective search and click trails [8] could yield interesting insights about formulating better queries. Finally, combining source retrieval with "online" text alignment of a downloaded candidate could help to ignore a lot of queries (and downloads) for passages that were already identified as aligned with a current candidate before the next query is submitted—opening new ways of query/download scheduling not possible with the current PAN source oracle.

REFERENCES

[1] Elizalde: Using statistic and semantic analysis to detect plagiarism. PAN 2013.
[2] Elizalde: Using noun phrases and tf-idf for plagiarized document retrieval. PAN 2014.
[3] Gillam, Newbold, Cooke: Educated guesses and equality judgements: Using search engines and pairwise match for external plagiarism detection. PAN 2012.
[4] Gollub, Stein, Burrows, Hoppe: TIRA: Configuring, executing, and disseminating information retrieval experiments. TIR@DEXA 2012.
[5] Grossman, Cormack, Roegiest: TREC 2016 Total Recall track overview.
[6] Haggag, El-Beltagy: Plagiarism candidate retrieval using selective query formulation and discriminative query scoring. PAN 2013.
[7] Hagen, Potthast, Stein: Source retrieval for plagiarism detection from large web corpora: Recent approaches. CLEF 2015.
[8] Hagen, Potthast, Völske, Gomoll, Stein: How writers search: Analyzing the search and writing logs of non-fictional essays. CHIIR 2016.
[9] Hagen, Stein: Candidate document retrieval for web-scale text reuse detection. SPIRE 2011.
[10] Han: Submission to the 7th competition on plagiarism detection. PAN 2015.
[11] Jayapal: Similarity overlap metric and greedy string tiling. PAN 2012.
[12] Kong, Han, Han, Yu, Wang, Zhang, Qi: Source retrieval based on learning to rank for plagiarism detection. PAN 2014.
[13] Kong, Lu, Han, Qi, Han, Wang, Hao, Zhang: Source retrieval and text alignment corpus construction for plagiarism detection. PAN 2015.
[14] Kong, Qi, Du, Wang, Han: Approaches for source retrieval and text alignment of plagiarism detection. PAN 2013.
[15] Kong, Qi, Wang, Du, Wang, Han: Approaches for candidate document retrieval and detailed comparison of plagiarism detection. PAN 2012.
[16] Lee, Chae, Park, Jung: CopyCaptor: Plagiarized source retrieval system using global word frequency and local feedback. PAN 2013.
[17] Maluleka: Derivative approach for plagiarism source retrieval. PAN 2016.
[18] Nourian: Submission to the 5th competition on plagiarism detection. PAN 2013.
[19] Potthast, Hagen, Beyer, Busse, Tippmann, Rosso, Stein: Overview of the 6th international competition on plagiarism detection. CLEF 2014.
[20] Potthast, Hagen, Stein, Graßegger, Michel, Tippmann, Welsch: ChatNoir: A search engine for the ClueWeb09 corpus. SIGIR 2012.
[21] Potthast, Gollub, Hagen, Graßegger, Kiesel, Michel, Oberländer, Tippmann, Barrón-Cedeño, Gupta, Rosso, Stein: Overview of the 4th international competition on plagiarism detection. CLEF 2012.
[22] Potthast, Gollub, Hagen, Tippmann, Kiesel, Rosso, Stamatatos, Stein: Overview of the 5th international competition on plagiarism detection. CLEF 2013.
[23] Potthast, Hagen, Völske, Stein: Crowdsourcing interaction logs to understand text reuse from the web. ACL 2013.
[24] Prakash, Saha: Experiments on document chunking and query formation for plagiarism source retrieval. PAN 2014.
[25] Rafiei, Mohtaj, Zarrabi, Asghari: Source retrieval plagiarism detection based on noun phrase and keyword phrase extraction. PAN 2015.
[26] Ravi N, Gupta: Efficient paragraph based chunking and download filtering for plagiarism source retrieval. PAN 2015.
[27] Stein, Meyer zu Eißen, Potthast: Strategies for retrieving plagiarized documents. SIGIR 2007.
[28] Suchomel, Brandejs: Heterogeneous queries for synoptic and phrasal search. PAN 2014.
[29] Suchomel, Brandejs: Improving synoptic querying for source retrieval. PAN 2015.
[30] Suchomel, Kasprzak, Brandejs: Three way search engine queries with multi-feature document comparison for plagiarism detection. PAN 2012.
[31] Suchomel, Kasprzak, Brandejs: Diverse queries and feature type selection for plagiarism discovery. PAN 2013.
[32] Veselý, Foltýnek, Rybička: Source retrieval via naïve approach and passage selection heuristics. PAN 2013.
[33] Williams, Chen, Chowdhury, Giles: Unsupervised ranking for plagiarism source retrieval. PAN 2013.
[34] Williams, Chen, Giles: Supervised ranking for plagiarism source retrieval. PAN 2014.
[35] Zubarev, Sochenkov: Using sentence similarity measure for plagiarism source retrieval. PAN 2014.