# Plagiarism Candidate Retrieval Using Selective Query Formulation and Discriminative Query Scoring

## Notebook for PAN at CLEF 2013

Osama Haggag and Samhaa El-Beltagy

Text Mining Research Group, Center for Informatics Science,
Nile University, Cairo, Egypt
osama.haggag@gmail.com, samhaa@computer.org

**Abstract.** This paper details the approach of implementing an English plagiarism source retrieval system to be presented at PAN 2013. The system uses the TextTiling algorithm to break a given document into segments that are centered around certain topics within the document. From these segments, keyphrases are generated using the KPMiner keyphrase extraction system. These keyphrases and segments are then used in generating queries indicative of the segment, and consequently the document. The queries are submitted to ChatNoir for finding plagiarism sources in the ClueWeb09 corpus from which the pan13 dataset is plagiarized. The target is to lessen the overall search effort while maximizing the performance by scoring unconsumed queries against the already downloaded candidate sources. Comparison to other PAN 2013 submissions for the same task, show the presented system to be one of the top performers.

## 1 Introduction

Plagiarism is the act of copying or using other people's data without permission or giving credit. It poses an unfortunate problem in modern academia as figures, charts, and even text from research papers and proposals can be copied without permission. For example, when asked to write a study on a certain topic, most students would start by searching the internet, or looking up Wikipedia. Some would then proceed to copy data on the subject, be it charts figures, tables or text, while *omitting* references to the sources. Such activities are deemed dishonorable and unacceptable in the academic community, and are considered as theft. It is often up to the reviewers of the material to detect the plagiarism through experience, and properly act upon it by proving and reporting it. Many tools have been developed recently for checking papers for plagiarism examples of which include Turnitin's OriginalityCheck[1] and iThenticate[2]. These tools check submitted papers for plagiarism by comparing them against a vast collection of documents in the form of webpages, papers, etc. The increasing demand for these tools serves to prove how severe the problem of plagiarism is becoming.

In this paper, we present an approach for generating discriminative queries from documents suspected of plagiarism. These queries are then used to search for possible sources of a suspect document at hand. This is done by dissecting the given document into related subtopics that are then condensed into characteristic keyphrases. The keyphrases are used to direct the focus of the search towards the possible sources of the aforementioned subtopics to avoid looking up as many irrelevant documents as possible.

The rest of this paper is organized as follows; **Section 2** describes the problem and explores its various dimensions. **Section 3** provides the details of the implementation. **Section 4** presents the performance of the system in comparison to other PAN 2013 system submissions. **Section 5** concludes the paper, and discusses possible future improvements.

## 2 Problem and Task Descriptions

### 2.1 Problem Description

Text plagiarism is a very common form of plagiarism, and is one of the prime focuses of PAN[3]. Unfortunately, detecting this kind of plagiarism can be a difficult task. The difficulty in detecting such plagiarism stems from the fact that it is relatively easy for one to conceal the plagiarism in a number of ways. One could obfuscate, paraphrase, or simply rearrange words, rendering it much harder for a machine, or even a human to detect the plagiarism. However, most plagiarists do not exert a lot of effort in their plagiarism as they simply copy and paste data.

There are two types of plagiarism detection, **extrinsic** and **intrinsic**[4]; intrinsic detection is the detection of deviations in the overall document style given only the document, thus identifying possible non-authentic bodies of text. Extrinsic detection is the task of identifying possible plagiarism instances in the document with relation to other documents. Extrinsic plagiarism detection is the focus of this work. Extrinsic plagiarism detection as a task is comprised of two sub-tasks: **source retrieval**, and **text alignment**. Source retrieval is the process of locating as many as possible source documents from which a suspect document has been plagiarized. Text alignment is the process of mapping passages within a suspect document to passages from which they have been copied in the source document. Source retrieval is the focus of this work.

### 2.2 PAN task Description

Participants in the source retrieval task were given a plagiarized dataset[5] that consists of suspicious documents. Each document addresses a certain topic and is plagiarized from web pages on that topic from the ClueWeb09[6] corpus. Analysis of the dataset revealed that there is little to no obfuscation in the documents. Some small passages and headlines are authentic and not plagiarized and the documents are well written, punctuated, and organized into paragraphs focusing on certain aspects.

The task requires processing the suspect documents to formulate queries that are indicative and characteristic of their content. These queries should then be used to search for the plagiarism sources in the ClueWeb09 corpus using one of two provided search engines: ChatNoir[7] or Indri[8].

Four different measures are used to assess any source retrieval system: Retrieval performance represented by Precision, Recall, and the F1 score, the workload represented by the number of queries and document downloaded, time till first detection, represented by the number of queries and documents downloaded before an actual source is located, and system runtime. The goal is to maintain a good balance between those four measures while aiming to maximize retrieval performance and to minimize workload and system runtime. With these constraints in mind, keeping a keen eye on downloads is determined to be the core of the problem. Downloading irrelevant documents would lead to more searching, possibly retrieving more documents that are irrelevant, and damaging the performance. Whereas downloading relevant documents would help minimize the search effort and sharpen the system's precision. As such, it was concluded that the strategy to be adopted must be devised to control the number of downloads.

Another factor that should be taken into consideration is that the ClueWeb09 corpus contains "spam" pages, and noisy documents such as huge site directories and listings. These could be wrongfully flagged as candidate sources, due to these pages having a huge variety of unrelated words that could trick the system into flagging them as a source of a certain query slowing the time to the first actual correct result, while dealing a blow to the precision of the system. Thus, we have also concluded that after retrieving candidate sources, and to keep the number of downloads and precision in check, there needs to be some kind of relation between the current downloaded candidates and the queries that have not been utilized yet. It would be favorable to score the unutilized queries against the already downloaded candidates to prevent the system from over-searching.

## 3 Implementation

The slight obfuscation in the dataset was determined not to affect the general outcome of the plagiarism detection process as initial experimentation showed that it did not hinder the searching process from retrieving documents that are relevant to the topic at hand. We have chosen ChatNoir as the search engine to use for searching for possible document sources. Within the proposed system, a number of phases were implemented. In the following subsections, each phase is detailed.

### 3.1 Preparing the Data

Given an input suspect plagiarized document, it is preprocessed with regular expressions to remove all non-English characters (including numbers, symbols, punctuation, etc) since the dataset is already known to be in English. It is then tokenized on whitespace, and the frequency distribution of the document is calculated. Then using the TextTiling[9] algorithm (provided by Python's NLTK[10] platform),

the document is divided into topically related **segments/subdocuments**. This step is tuned to produce a relatively small number of large segments, as explained in subsection 3.4.

### 3.2 Formulating the Queries

Keyphrases are extracted for each of the document's generated **segments** using the KPMiner[11] keyphrase extraction system, which returns the topmost keyphrase of the segment. This keyphrase (which is a phrase of 1 to 3 words on average) is said to be characteristic of the segment. The segment is then divided into sentences, and every four sentences are grouped into a **chunk**. A query is then generated for each chunk.

Each chunk is preprocessed exactly like the main document, and tokenized on whitespace. English stopwords are removed, and the unique words (words with a frequency of 1 over the document) are identified. These unique words are removed from the chunk, and added to the chunk's query. The chunk now contains no stopwords, or unique words. The remaining non-unique words in the chunk are then sorted by ascending frequency, and moved into the query in their sorted order.

The query is now organized in such a way that the unique words are at the beginning, and then the rest of the words in ascending frequency. If the length of the query is greater than 10 words, the query is trimmed down to 10 keywords taking only the first 10 terms that appear in the query. This is due to ChatNoir's keyword limit of 10. The segment's keyphrase (which could be of length n-words) is then added in place of the last n words in the query if the query does not already contain the keyphrase's terms. The queries are stored as a list of strings per document. An overview of the process of query generation is shown in Fig.1.

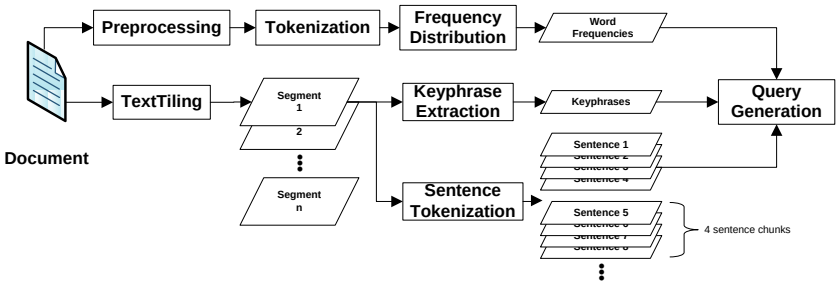


Fig. 1, Overview of query generation

### 3.3 Searching

ChatNoir allows for requesting snippets of the search results of a query. A snippet request returns a number of characters (500 characters) from the search result around the terms of the submitted query. Snippets do not count as "downloads" when calculating performance. For each document, its list of queries is traversed as follows:

The first query is submitted to ChatNoir for a snippet request. The query is scored against the snippet through simple token matching; if 50% or more of the query's tokens are found in the 500-character snippet's tokens, then the search result (the

snippet's origin) is considered a candidate source. This candidate is then downloaded and added to a list of downloaded documents related to the document at hand.

Each of the subsequent queries is then checked against the list of downloaded documents through simple token matching as well. If 60% or more of the query's tokens are found in any of the downloaded documents' tokens, then this document is considered a source for this query, and this query will not be used for searching. If the query fails to score 60% against all the downloaded documents, it is then used for a new snippet request, and possibly to download a new candidate source as above. The algorithm for scoring the queries is illustrated in pseudo-code below:

```
For i = 1 to Length(queries): #1st query has been used
    For j = 0 to Length(downloadedDocs):
        If ScoreQuery(queries[i], downloadedDocs[j]) > 0.6:
             FoundCandidate = True;
            Break; #Found possible source for this query
    If not (FoundCandidate):
        Snippet = GetSnippet(queries[i])
        If ScoreSnippet(Snippet, queries[i]) > 0.5:
            downloadedDocs.add(Download(Snippet.id));
```

When downloading documents, ChatNoir provides an "oracle" function. Given the id of the suspicious document in the training set that you are searching for, and the id of the page you are trying to download, the oracle would inform you if the downloaded page is a source for the suspicious document or not. This is used to calculate precision, recall, and the time to the first actual result.

### 3.4 Tunable Parameters

The plagiarism detection system detailed above is affected by a number of tunable parameters. Due to the time cost of the runs over the dataset, as a typical run takes around 2-3 hours, it was decided that determining the optimum factors through iteration over **all** the different combinations of the parameters would take a lot of time. As a result, a different approach was used to obtain values that would be good enough, but are not necessarily globally optimal. Through human intuition and common sense, and a small number of experiments we tried to find an optimum value for each of the parameters. Below, we go through the reasoning behind the various configurations. The row in bold in the tables denotes the submitted configuration.

**TextTiling parameters:** TextTiling offers control over how large the subdocuments/segments can be, by changing two parameters **w**, and **k**; TextTiling is done by tokenizing the text into pseudo-sentences of a fixed size **w**, where **k** is the size (in sentences) of the block used in the block comparison method[12]. The size of the pseudo-sentences control to a large degree the number of segments generated.

Through experimentation, it was found that tuning the segmentation for a large number of topics of small size leads to higher recall but lesser precision. This is logical due to there being a lot of variance in the queries due to the large number of topics and their different keyphrases; Many queries would fail to score 60+% on the downloaded documents due to the variety in the downloads. This leads the system to carry out more searching and downloading, which in the end damages the precision, without much gain in recall as shown in Table 1.

It is determined that setting the segmentation to collect larger topics (an average of 15 segments per document) is best for both precision and recall. This is obtained by setting **w = 50**, and **k = 5** as can be seen in Table 1.

| | *No. of Topics* | *Precision* | *Recall* | *No. Qrs* | *No. Dlds* | *1st Detection* |
|---|---|---|---|---|---|---|
| w=20, k=5 | 35 | 0.67 | 0.44 | 42.9 | 7.85 | 12.55 |
| **w=50, k=5** | **15** | **0.72** | **0.44** | **36.33** | **7.4** | **9.775** |
| w=80, k=30 | 11 | 0.69 | 0.43 | 32.15 | 7.45 | 11.1 |

Table 1, TextTiling parameters w, k at a Snippet score of 50%, and a Query score of 60%, Chunk Size of 4, Frequency Threshold of 1. First Detection is in queries. All numbers are averages over the dataset

**Chunk size selection**: there is a choice as to how large the **sentence chunk size** could be. Setting the chunk size to 1 (one sentence per chunk, i.e., one sentence per query) leads to more searching, and higher recall at a loss for precision. The choice of four sentences was determined by running a number of experiments to determine the best performing chunk size and as can be seen from table 2, a chunk size of 4 was best for both precision and recall. There is also the choice of the **frequency threshold** that identifies "unique" words. Several values were also tested in the same manner and a threshold of 1 was found to be best as shown in Table 2.

| | *Precision* | *Recall* | *No. Qrs* | *No. Dlds* | *1st Detection* |
|---|---|---|---|---|---|
| CS=1, FT=1 | 0.45 | 0.5 | 53 | 13 | 12 |
| CS=3, FT=1 | 0.65 | 0.46 | 37.85 | 8.8 | 9.7 |
| **CS=4, FT=1** | **0.72** | **0.44** | **36.33** | **7.4** | **9.775** |
| CS=5, FT=1 | 0.71 | 0.41 | 28.6 | 6.7 | 8.7 |
| CS=4, FT=3 | 0.73 | 0.4 | 30.45 | 7.0 | 8.95 |
| CS=4, FT=5 | 0.63 | 0.37 | 27.25 | 6.85 | 11.35 |

Table 2, Choice of sentence chunk size (CS) and the frequency threshold (FT) at w=50, k=5, Snippet Score of 50%, Query score of 60%. First Detection is in queries. All numbers are averages over the dataset

**Search parameters:** one can retrieve more than one result for a certain query. Doing so does not benefit the performance, as the first result is often the most correct one. There are also the two scoring functions: one that scores **queries against snippets**, and **queries against candidate downloaded documents**. The snippet score is chosen to be 50%. On trying higher values, more snippets would fail to pass the check due to their limited number of characters, damaging both the precision and the recall. For lower values most snippets would pass the check flagging more documents for downloads, damaging the precision. The same rationale goes for scoring queries against the candidate documents as in shown Table 3.

| ***Snippet Score*** | ***Query Score*** | ***Precision*** | ***Recall*** | ***No. Qrs*** | ***No. Dlds*** | ***1st Detection*** |
|---|---|---|---|---|---|---|
| 40 | 60 | 0.71 | 0.45 | 35.675 | 7.5 | 9.55 |
| **50** | **60** | **0.72** | **0.44** | **36.33** | **7.4** | **9.775** |
| 60 | 60 | 0.72 | 0.42 | 37.65 | 7.05 | 10.425 |
| 50 | 40 | 0.75 | 0.4 | 29.65 | 6.175 | 9.625 |
| 50 | 70 | 0.72 | 0.45 | 39.5 | 7.525 | 9.75 |

Table 3, Choice of Snippet and Query scores at w=50, and k=5, chunk size of 4, Frequency Threshold of 1. First Detection is in queries. All numbers are averages over the dataset

## 4 Results

The presented system was evaluated using the four measures described in section 2.2 in addition to the "No Detection" metric. Our system was determined to be one of the top three submissions to PAN'13 in the source retrieval task. Among the top three participants (shown in Table 4a), our system has the **highest precision**, the **lightest workload** and the **fastest runtime**. Our system is also the **fastest system to download the first source**. Overall, our system has the highest number of top performance indicators. In addition, our system has the **second-highest** recall and F1 score. Table 4b provides a comparison between the performance of our system and the other participants of PAN13.

| | **Retrieval Performance** | | | **Workload** | | **1st Detection** | | **No Detection** | **Runtime** |
|---|---|---|---|---|---|---|---|---|---|
| | *F1* | *Prec* | *Recall* | *Qrs* | *Dlds* | *Qrs* | *Dlds* | | |
| **Haggag** | 0.44 | **0.63** | 0.38 | **32.04** | **5.93** | 8.92 | **1.47** | 9 | **9162471** |
| Williams | **0.47** | 0.55 | **0.50** | 116.4 | 14.05 | 17.59 | 2.45 | **5** | 69781436 |
| Lee | 0.35 | 0.50 | 0.33 | 44.04 | 11.16 | **7.74** | 1.72 | 15 | 18628376 |

| **Submission** | **Retrieval Performance** | | | **Workload** | | **Time to 1st Detection** | | **No Detection** | **Runtime** |
|---|---|---|---|---|---|---|---|---|---|
| | $F_1$ | Precision | Recall | Queries | Downloads | Queries | Downloads | | |
| elizalde13 | 0.17 | 0.12 | 0.44 | 44.50 | 107.22 | 16.85 | 15.28 | **5** | 14504695 |
| foltynek13 | 0.15 | 0.11 | 0.35 | 161.21 | 81.03 | 184.00 | 5.07 | 16 | 39317468 |
| gillam13 | 0.04 | 0.02 | 0.10 | 16.10 | 33.02 | 18.80 | 21.70 | 38 | **906327** |
| haggag13 | 0.44 | **0.63** | 0.38 | 32.04 | **5.93** | 8.92 | **1.47** | 9 | 9162471 |
| kong13 | 0.01 | 0.01 | **0.65** | 48.50 | 5691.47 | 2.46 | 285.66 | 3 | 245882767 |
| lee13 | 0.35 | 0.50 | 0.33 | 44.04 | 11.16 | 7.74 | 1.72 | 15 | 18628376 |
| nourian13 | 0.10 | 0.15 | 0.15 | **4.91** | 13.54 | **2.16** | 5.61 | 27 | 1516482 |
| suchomel13 | 0.06 | 0.04 | 0.23 | 12.38 | 261.95 | 2.44 | 74.79 | 10 | 98274058 |
| williams13 | **0.47** | 0.55 | 0.50 | 116.40 | 14.05 | 17.59 | 2.45 | **5** | 69781436 |

Table 4a (top), Performance comparison with the top three participants. Table 4b (bottom), Performance of all participants. In bold are the metrics in which each system performed better compared to the others

## 5 Conclusion and Future Work

This paper has presented a system that can retrieve plagiarism sources while minimizing the workload. The system is capable of achieving its goal by careful formulation and elimination of queries to be submitted for search. The system utilizes two algorithms in generating the queries, namely the TextTiling algorithm, and the KPMiner keyphrase extraction algorithm as well as a set of heuristics for employing those. The system's performance was shown to be among the best this year, as shown in Table 4.

There is however room for improvement on the current system. For example, the values of the used parameters could be further optimized. Measures could also be taken to counter the obfuscation without adding too much complexity to the system. More use of ChatNoir's additional functionality can be made. This includes for example, making use of ChatNoir's batch query service where you can request search results for more than one query in the same request. This can be used on the document level, by coming up with a method of speeding up the searching process by using more than one query at the same time, while maintaining the scoring scheme. It can

also be used on a dataset level basis by processing a number of documents simultaneously. Moreover, ChatNoir provides some advanced parameters for searching, such as defining spam rank, page rank limits, and a proximity factor between queries and the results they return. All these could be used to refine the search results, filtering out unwanted result pages. The scoring functions could take into consideration the context of the query while scoring queries against candidate documents instead of simple token matching. The code to our implementation will be available under the MIT license[13] for whoever wants to extend or use our system.

## References


1. OriginalityCheck, http://turnitin.com/en_us/products/originalitycheck.
2. iThenticate, http://www.ithenticate.com/.
3. PAN 2013, http://pan.webis.de/.
4. Potthast, M., Gollub, T., Hagen, M.: Overview of the 4th International Competition on Plagiarism Detection. Pamela Forner, Jussi …. 17–20 (2009).
5. Potthast, M., Hagen, M., Völske, M., Stein, B.: Crowdsourcing Interaction Logs to Understand Text Reuse from the Web. 51st Annual Meeting of the Association of Computational Linguistics (ACL 13) (2013).
6. ClueWeb09 Dataset, http://lemurproject.org/clueweb09/.
7. Potthast, M., Hagen, M., Stein, B., Graßegger, J., Michel, M., Tippmann, M., Welsch, C.: ChatNoir: A Search Engine for the ClueWeb09 Corpus. In: Hersh, B., Callan, J., Maarek, Y., and Sanderson, M. (eds.) 35th International ACM Conference on Research and Development in Information Retrieval (SIGIR 12). p. 1004. ACM (2012).
8. Indri, http://www.lemurproject.org/indri/.
9. Hearst, M.: TextTiling: Segmenting text into multi-paragraph subtopic passages. Computational linguistics. 23, 33–64 (1997).
10. Natural Language Toolkit, http://nltk.org/.
11. El-Beltagy, S.R., Rafea, A.: KP-Miner: A keyphrase extraction system for English and Arabic documents. Information Systems. 34, 132–144 (2009).
12. NLTK's TextTiling Module Documentation, http://nltk.org/_modules/nltk/tokenize/texttiling.html.
13. Haggag, O., El-Beltagy, S.R.: Plagiarism Candidate Retrieval System, https://github.com/osama-haggag/Plagiarism-Source-Retrieval-for-PAN13