**Available Online at www.ijcsmc.com**

**International Journal of Computer Science and Mobile Computing**

**A Monthly Journal of Computer Science and Information Technology**

**ISSN 2320–088X**
**IMPACT FACTOR: 5.258**

***IJCSMC, Vol. 5, Issue. 3, March 2016, pg.81 – 85***

# Efficient Technique to Retrieve Plagiarized Documents for Plagiarism Detection

**G.Srikanth Reddy[1], P. Jeevan Kumar[2]**
[1]Associate Professor, Dept of IT, Vardhaman College of Engineering
[2]Assistant Professor, Dept of IT, Vardhaman College of Engineering
[1] srikanthreddygopu@gmail.com, [2] jeevankumar922@gmail.com

***Abstract. This paper details the approach of implementing an English plagiarism source retrieval system. A given document is broke down into segments by using TextTiling algorithm. These segments , are centered around certain topics within the document, key phrases are generated using KPMiner keyphrase extraction system. Segments and key phrases are used to create queries of the segment and document. ChatNoir search engine is used to find plagiarism from the above queries once we submit our queries to the search engine. This paper helps in improving the performance with less effort by scoring unconsumed queries against the already downloaded candidate sources. This approach is one of the top approach when compared with all other detection approaches***

***Keywords: extrinsic plagiarism detection, source retrieval, TextTiling algorithm, chunking, ChatNoir search engine***

## 1. Introduction

Plagiarism is to copy the data of other person(s) without giving credit. This is a problem now a days, copying text, figures and charts from proposals and research papers without permission of the respective author who wrote the paper. Most of the researchers or students who start their study on a certain topic will search in the internet or looking into Wikipedia. Some would copy content from the internet with out any proper citation. Such activity is considered as unacceptable and dishonorable in the research and academic community treating as theft. Reviewers have to detect the plagiarism with their experience and report it as plagiarized. Some of the tools available to check Plagiarism are iThenticate[2], Turnitin and Originality Check[1]. These tools check the Plagiarism and comparing the source document with a vast collection of documents in the form of web, papers etc.

In this paper, the approach is to generate different queries from the document to be checked for Plagiarism. From the queries generated, the sources (if any) of a suspect document are found. This source retrieval is done by dividing the document into subtopics that are reduced into characteristic keyphrases. These keyphrases are used to search for the possible source of above mentioned subtopics and avoid looking into documents which are not related

The rest of this paper is organized as follows; **Section 2** problem description and explores its various dimensions.

**Section 3** providing implementation details. **Section 4** performance of the system presentation. **Section 5** concludes the paper, and discusses possible future improvements.

## 2. PROBLEM AND TASK DESCRIPTIONS

### *2.1 Problem Description*

One of the prime focus and common form of plagiarism is text plagiarism [3] and this form of plagiarism detection is a difficult task. As it is a text one can paraphrase, rearrange words and obfuscate the content of the original document, makes it difficult and hard for a machine to detect plagiarism.

Plagiarism is of two forms, intrinsic and extrinsic [4], intrinsic plagiarism is to find the deviations in the overall document style with in it. Extrinsic plagiarism is a combination of source retrieval and text alignment. Source retrieval is the process of identifying all the source documents available in the corpus from which the suspect document is copied. Text alignment is to map passages from suspect document to passages from which they were copied in the source document. This paper focuses on source retrieval.

### *2.2 Task Description*

Source retrieval task is given plagiarized dataset [5] consisting of suspicious documents. All the documents are plagiarized from the web on their respective topic from ClueWeb09 [6] corpus. Analyzing the dataset revealed that there is no obfuscation to little obfuscation in the documents. The documents are punctuated, written well and organized into paragraphs, some small passages, headlines are genuine and not plagiarized focusing on certain aspects. The task is of processing the suspect documents and formulating 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: Indri[8] or ChatNoir[7].

Retrieval performance is measured using 3 measures: Precision, Recall and the F1 measure. The target is to increase the retrieval performance by maintaining balance between the above measures and minimizing the system runtime, workload. Downloading irrelevant documents damages the performance and downloading relevant documents help minimize search effort improving the precision. Here the strategy is 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. ChatNoir search engine is used for searching 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, preprocessing is done using regular expressions to remove all non-English characters (including symbols, numbers, punctuation, etc) since the dataset is already known to be in English. It is then tokenized considering whitespace, and the frequency distribution of the document is calculated. The document is divided into topically related **segments/subdocuments** using the TextTiling algorithm (provided by NLTK[10] platform). Tuning the results produces a relatively small number of large segments, as explained in subsection 3.4.

### *3.2 Formulating the Queries*

Using the segments generated in the above phase keyphrases are extracted using the KPMiner[11] keyphrase extraction system. Each segment returns the topmost keyphrase in it as a result from the KPMiner. This phrase, the characteristic of the segment is in the length of 3 to 4 words. The segment is divided into sentences and every four sentences are grouped into a chunk, using each chunk a query is prepared. Each chunk is preprocesses like the main document. Stopwords are removed and unique words are identified. These unique words are added to the query. Now the chunk doesn't have any stopwords or unique words. The remaining words in the chunk are then arranged in the sorted ascending order and moved into the query in their sorted order. The query is now setup so that unique words are at the beginning and remaining words in their ascending order of frequency. As the ChatNoir search engine accepts a query of 10 words maximum, the length of query formed is checked and if it is greater than 10 words , the query is trimmed by considering only the first 10 words in the query. The segment's keyphrase, a length of n-words, is added in place of last n words in the query if the query is not already with keyphrase's terms. Queries are stored as list of strings per document. Overview of the process of query generation is shown in the fig.1.

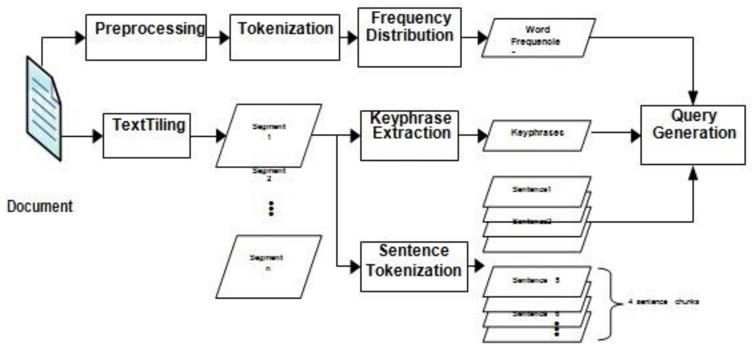

Fig. 1, Overview of query generation

### *3.3 searching*

A snippet is prepared with 500 characters, which is brought from the search results of a query. Snippets are not counted as "downloads" while calculating performance. For each document the queries are traversed as follows. First query is given to the search engine to get the snippet. Now the query is scored against the snippet by simple token matching. If the match is 50% or more , matching is done by checking how many tokens in the query are available in 500 character snippet tokens, then the result is considered as candidate score. This document is then downloaded and added to list of documents downloaded related to the suspicious document. Simple token matching is performed on the list of downloaded documents by the subsequent queries. In any of the downloaded document if 60% or more of the query's tokens are found then the document is considered as source for this query, this query is not considered for searching. If the match is below 60% against all downloaded documents, it is used for a new snippet request and a possible download as new candidate source.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. A run consumes 2-3 hours and tried to identify optimum factors by iterating over all different combinations of the parameters but couldn't improve the performance so a different approach was used to get values which are good but not globally optimal. Some experiments were conducted manually to find an optimum value for each of the parameters. Below are the reasons to consider various configurations. In the table, row highlighted in bold letters is the submitted configuration.

**TextTiling parameters:** TextTiling controls over segments/ subdocuments size like how large it can be by considering two parameters w, k. TextTiling is generate tokens from the text by creating pseudo sentences of a fixed size **w**, and **k** is the size( in sentences) of the block used in block comparison method[12]. The size of pseudo sentences controls the number of segments generated. By experimenting it was understood that tuning the segmentation for a large number of small sized topics leads to higher recall but lesser precision. This is logical because there is a lot of variance in queries as there are large number of topics and their different keyphrases; many queries fail to get a score of 60%+ on downloaded documents due to variety in downloads. This leads the system to carry out more searching and downloading, resulting in damaging precision without improving recall , 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. Dids |
|---|---|---|---|---|---|
| W=20, k=5 | 34 | 0.66 | 0.43 | 42.7 | 7.89 |
| W=50, k=5 | 16 | 0.71 | 0.45 | 36.32 | 7.5 |
| W=80, k=30 | 10 | 0.68 | 0.43 | 32.16 | 7.46 |

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. All numbers are averages over the dataset

**Chunk size selection**: chunk size selection plays a crucial role in the retrieval. Having a chunk size of 1 (one sentence per query or one chunk per sentence) takes more time to search, and produces higher recall leading to loss for precision. After running a number of experiments choice of having 4 sentences gives the best performing chunk size and can be seen in table 2, this gives the best precision and recall. The choice of the frequency threshold that identifies "unique" words was also identified in the same manner. Several values were also tested for frequency threshold and a threshold of 1 was found to be best as shown in Table 2.

| | Precision | Recall | No. Qrs | No. Dids |
|---|---|---|---|---|
| CS=1, FT=1 | 0.46 | 0.5 | 53 | 13 |
| CS=3, FT=1 | 0..64 | 0.46 | 37.85 | 8.8 |
| CS=4, FT=1 | 0.72 | 0.44 | 36.33 | 7.4 |
| CS=5, FT=1 | 0.71 | 0.41 | 28.6 | 6.7 |
| CS=4, FT=3 | 0.73 | 0.4 | 30.45 | 7.0 |
| CS=4, FT=5 | 0.63 | 0.37 | 27.24 | 6.84 |

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%. All numbers are averages over the dataset

**Search parameters:** A query may give more than one result which doesn't improve the performance. Two scoring functions are considered: one that scores queries against candidate downloaded documents & the other queries against snippets. Choosing higher values damages the precision and recall because number of characters in snippet is limited and would fail the pass check. Choosing a lower value helps in passing the check but results in downloading more documents losing the precision, so a snippet score of 50% is chosen. The same is used for scoring queries against the candidate documents as in shown Table 3.

| Snippet Score | Query Score | Precision | Recall | No. Qrs | No. Dids |
|---|---|---|---|---|---|
| 40 | 60 | 0.71 | 0.45 | 35.675 | 7.5 |
| 50 | 60 | 0.72 | 0.44 | 36.33 | 7.4 |
| 60 | 60 | 0.72 | 0.42 | 37.65 | 7.05 |
| 50 | 40 | 0.75 | 0.4 | 29.65 | 6.175 |
| 50 | 70 | 0.72 | 0.45 | 39.5 | 7.525 |

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

## 4. Results

The three measures are used to evaluate the presented system. Our system generated best results in the source retrieval task. Our system has the **fastest runtime** and **highest precision**. Overall, our system has the highest number of top performance indicators.

| | Retrieval Performance | | |
|---|---|---|---|
| | F1 | Precision | Recall |
| Accuracy | 0.44 | 0.63 | 0.38 |

Table 4. Performance of our approach

## 5. Conclusion and Future Work

This paper has presented a system that minimizes the workload while retrieving plagiarism sources. Two algorithms are used to generate queries, TextTiling and KPMiner keyphrase extraction algorithm and some heuristics. This system improved the performance by careful formulation and elimination of queries which are to be submitted for search. In future this system can be improved by optimizing the values of the used parameters; obfuscation can be avoided without much complexity, use of the additional functionality in ChatNoir like batch query service which gives results for more than one query in the same request. Using the additional functionality of ChatNoir helps in speeding up searching process as takes multiple queries as a batch and number of documents can be processed simultaneously. Some other features of ChatNoir like spam rank, page rank limits, proximity factor between queries can be used as advanced parameters for searching helping in refining search results, filtering unwanted result pages. Scoring functions can be considered to score queries against candidate document instead of simple token matching.

## 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