Derivative Approach for Plagiarism Source Retrieval

Rhulani Maluleka

Peoples' Friendship University of Russia, Moscow, Russia
rhumaluleka@gmail.com

Abstract. *This paper describes our approach to the PAN shared task of plagiarism source retrieval based on the strategy suggested by Williams et. al [1]. We also incorporate named entities queries similar to those of Elizalde [2].*

1 Algorithm

We attempt to improve on the current approaches to the source retrieval task. Based on the results of the 2015 competition [3] we chose to implement an algorithm that combines the approaches of Williams et. al [1] and Elizalde [2]. Williams' software was the best performing detector in the source retrieval task in 2013 and 2014. Even though they did not submit a new version in 2015, thier approach still went unmatched by the 2015 participants. We chose to work from their 2013 approach as the added complexity of their supervised result ranking strategy in 2014 achieved virtually the same results as its predicessor. Elizalde makes use of a novel idea of extracting named entities across each document in an attempt to detect highly obfuscated plagiarism.

Our approach consists of several stages, namely: chunking, key-phrase extraction, query formulation, and download filtering.

Chunking: We begin by segmenting the text of the suspicious document into paragraphs of 5 sentences each. We preprocess each paragraph, removing all non-alphabetic characters.

Keyphrase Extraction and Query Formulation: In forming keyphrases two distinct methods were employed. The first attempt to find the most important features of the entire document, while the other forms queries based on individual chunks.

Named entity queries: The Natural Language Toolkit (NLTK)[1] is used to identify Named Entities over the whole text. These are then ranked in descending order of length. The 10 longest named entities are submitted as-is as queries to the search engine. As noted by Elizalde [2], the rationale behind is that the named entities are unlikely to change even if paraphrasing has been used to obfuscate plagiarism. Additionally, the longer named entities are likely to contain more specific information, and thus be more likely to yield true positive results.

[1] nltk 3.2.1 http://www.nltk.org/

Chunk based queries: Each sentence in each paragraph is tokenized using NLTK's Punkt Sentence Tokenizer and the NLTK pre-trained part-of-speech tagger is used to tag all the tokens. All stopwords remove, and only verbs, nouns, and adjectives are retained. The WordNet lemmatizer is used to stem word. Stemming is done last as it may affect the identification of named entities. Queries are formed by concatenating sequences of tokens to form disjunct sequential 10-grams. The first three 10-grams from each paragraph are submitted to the ChatNoir search engine. The top three results are returned for each queries.

Download Filtering: The ChatNoir search engine [4] allows one to request a snippet, of up to five hundred characters, of a specific document. The snippet is based around a query that is submitted along with the request. We use the original query that returned the result is used for requesting snippets; and request snippets of the maximum length. Documents are only downloaded if they are deemed similar to the suspicious document, based on their snippet. The similarity between the snippet of each document and the suspicious document is calculated based on an method suggested by Broder et. al [5] using word n-grams, also known as shingles. The 5-shingles, overlapping sequences of 5 tokens, are extracted from the document and each downloaded snippet. The similarity between a snippet s and a suspicious document d is then calculated as:

$$Sim(s, d) = S(s) \cap S(d)$$

where S is a set of shingles. The results are then ranked by the similarity measure of their snippet.

The figure (see Algorithm 1) shows the algorithm for our source retrieval approach. The implementation of our algorithm is publicly available through PAN's online code repository.[2]

References

1. Williams, K., Chen, H.-H., Choudhury, S. R. & Giles, C. L. Unsupervised Ranking for Plagiarism Source Retrieval. *Notebook for PAN at CLEF 2013* (2013).
2. Elizalde, V. *Using statistic and semantic analysis to detect plagiarism* in *CLEF (Online Working Notes/Labs/Workshop)* (2013).
3. Hagen, M., Potthast, M. & Stein, B. Source Retrieval for Plagiarism Detection from Large Web Corpora: Recent Approaches. *Working Notes Papers of the CLEF,* 1613–0073 (2015).
4. Potthast, M., Hagen, M., Stein, B., Graßegger, J., Michel, M., Tippmann, M. & Welsch, C. *ChatNoir: A Search Engine for the ClueWeb09 Corpus* in *35th International ACM Conference on Research and Development in Information Retrieval (SIGIR 12)* (eds Hersh, B., Callan, J., Maarek, Y. & Sanderson, M.) (ACM, Aug. 2012), 1004.

[2] https://github.com/pan-webis-de/maluleka16

5. Broder, A. Z., Glassman, S. C., Manasse, M. S. & Zweig, G. Syntactic clustering of the web. *Computer Networks and ISDN Systems* **29,** 1157–1166 (1997).

Algorithm 1 Source Retrieval Approach

```
1: procedure SourceRet(text)
2:     NEs ← getNamedEntities(text)
3:     for all result in NEs do
4:         snippet ← getSnippet(result)
5:         if similarity(snippet) ≥ min_Sim then
6:             if (result ∈ sources) = False then
7:                 sources ← Download(result)
8:             end if
9:         end if
10:    end for
11:    paragraphs ← splitText(text)
12:    for all p in paragraphs do
13:        p ← preprocess(p)
14:        queries ← extractTopQueries(p)
15:        for all q ∈ queries do
16:            results ← submitQueries(q)
17:        end for
18:        results ← rankResults(results)
19:        for all result in results do
20:            snippet ← getSnippet(result)
21:            if similarity(snippet) ≥ min_Sim then
22:                if (result ∈ sources) = False then
23:                    sources ← Download(result)
24:                end if
25:            end if
26:        end for
27:    end for
28: end procedure
```