Using statistic and semantic analysis to detect plagiarism

Notebook for PAN at CLEF 2013

Victoria Elizalde

kivielizalde@gmail.com

Abstract This paper describes an approach submitted to the 2013 PAN competiton for the source retrieval sub-task. Three different methods for extracting queries were used, which employed tf-idf, noun phrases and named entities, in order to submit very different queries and maximize recall.

1 Introduction

To plagiarize is to take someone else's work or ideas and pose it as your own. It has become a major problem for Universities and other academic institutions since Internet has become widespread. Current plagiarism detection methods should check the whole Web to find possible matches. For this reason, since last year, the Plagiarism detection track in PAN[1] has been divided in two sub tasks: source retrieval and detailed comparison. This notebook reports an approach presented to the PAN 2013 plagiarism competition for the first sub-task.

2 Candidate retrieval

Source retrieval - sometimes called candidate document retrieval - is the first step of the plagiarism detection process. It consists in finding a set of documents which are likely to contain plagiarism, analyzing the document from a global perspective, either by using an index or querying a search engine. After this stage a second step is performed: detailed comparison, in which the previously retrieved documents are compared exhaustively against the suspicious document. Source retrieval is a recall oriented problem, since in the second step it is possible to increase the precision of the overall system, while sometimes lowering the recall[5].

This year the corpus utilized was ClueWeb09[7] and two different search engines were available to search it: ChatNoir[6] and Indri[8]. The ChatNoir engine only supports keyword search, while Indri has quite a complex query language grammar. Both engines were used in this work: ChatNoir for keyword queries and Indri where an exact match to a phrase was needed.

The approach used to solve this task consisted in three different strategies to find plagiarized texts, which will be discussed in the following subsections. It was developed using Python and the Natural Language Toolkit [2].

[1] pan.webis.de

2.1 Tf-idf based queries

This first strategy consisted in keyword based queries, submitted to the ChatNoir engine. The text was divided in 50 line chunks, non alphabetical characters and stopwords were removed. Lemmatization was applied using the WordNet lemmatizer[3] and words were ranked by their tf-idf coefficient. The list of frequency words used was generated using the Brown Corpus[4] also applying the afore mentioned preprocessing (stopword removal, WordNet lemmatization). Finally, a query with the top 10 ranked words was generated for each chunk.

2.2 Named Entity based queries

For this approach, NLTK was used to identify Named Entities which were ranked according to the amount of words included. The top 10 entities were submitted to Indri to search for an exact match. This yields at most 10 queries per document.

The rationale behind this is that even when there is some paraphrasing, the Named Entities (places, people, etc) will remain unchanged. Also, the longest NEs will be less common and hence appear in less documents.

2.3 Noun phrase based queries

Finally, an existing keyphrase extractor was adapted to the task of plagiarism detection. Barker and Cornacchia[1] search for noun phrases in the text, cluster them according to their head noun and select the n clusters which contain the most phrases. Each NP is then scored by multiplying the length of the phrase by the number of phrases which contains its head noun. The n best scored phrases are then kept.

In this work, the default NLTK POS tagger was used, and the noun phrases were found by using fixed patterns. With $m = 20$ and $n = 15$, this strategy generated at most 15 queries per document.

A slight modification to the algorithm was introduced: all the nouns present were used in the ranking, not just the head nouns. For example, in the phrase "the Church of Ireland", the phrase would count both towards "Church" and "Ireland".

The queries were posed to the Indri search engine.

2.4 Query combination

In all cases, only the top 10 results of every query were analyzed. For each result, a 160 character snippet was requested. The words were POS-tagged and only verbs, adjectives and nouns were considered. If more than 90% of those words (or their stemmed form) were present in the suspicious text, the document was regarded as promising and downloaded.

Table 1. PAN 2013 Source retrieval final results

Submission	Retrieval Performance			Workload		Time to 1st Detection		No	Runtime
	F_1	Precision	Recall	Queries	Downloads	Queries	Downloads	Detection	
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

3 Discussion

The goal behind using three different approaches of query extraction, with different chunk lengths is to generate different sets of queries, thus maximizing recall. This sacrifices precision. The reasoning behind this is that the second phase in plagiarism detection - detailed comparison - will improve performance, while recall won't be improved, but rather lowered. The results obtained in the competition clearly are a consequence of these decisions.

Since in some contexts queries are charged while downloads aren't, another decision made was to minimize the number of queries. For that reason, very large chuncks (50 lines) were used for the first strategy, while for the other strategies a fixed lower bound on the amount of queries was set (10 and 15 queries per document, respectively). However, a large number of documents (10) were downloaded for each query, to ensure recall was high.

When looking at the results, we can see that the average queries per document are 44.5, while the average downloads are 107.22. This yields approximately 2.4 downloads per query, which is far lower than 10. There are two reasons that can explain this: on one side, two of the strategies employ exact match searches, which typically result in fewer documents. On the other side, this could mean that filtering downloads using the text snippets lowers the number of downloaded documents dramatically.

References

1. Barker, K., Cornacchia, N.: Using noun phrase heads to extract document keyphrases (2000)
2. Bird, S., Klein, E., Loper, E.: Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit. O'Reilly, Beijing (2009), http://www.nltk.org/book
3. Fellbaum, C.: WordNet: An Electronic Lexical Database. Bradford Books (1998)
4. Francis, W.N., Kucera, H.: Brown corpus manual. Tech. rep., Department of Linguistics, Brown University, Providence, Rhode Island, US (1979), http://icame.uib.no/brown/bcm.html
5. Potthast, M., Gollub, T., Hagen, M., Kiesel, J., Michel, M., Oberländer, A., Tippmann, M., Barrón-Cedeño, A., Gupta, P., Rosso, P., Stein, B.: Overview of the 4th international competition on plagiarism detection. In: Forner, P., Karlgren, J., Womser-Hacker, C. (eds.) CLEF (Online Working Notes/Labs/Workshop) (2012)

6. 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., Sanderson, M. (eds.) 35th International ACM Conference on Research and Development in Information Retrieval (SIGIR 12). p. 1004. ACM (Aug 2012)
7. Potthast, M., Hagen, M., Völske, M., Stein, B.: In: 51st Annual Meeting of the Association of Computational Linguistics (ACL 13)
8. Strohman, T., Metzler, D., Turtle, H., Croft, W.B.: Indri: a language-model based search engine for complex queries. Tech. rep., in Proceedings of the International Conference on Intelligent Analysis (2005)