A Set-Based Approach to Plagiarism Detection
Notebook for PAN at CLEF 2012

Robin Küppers and Stefan Conrad

Institute of Computer Science, University of Düsseldorf, Germany
{kueppers, conrad}@cs.uni-duesseldorf.de

Abstract This paper describes our approach to the Detailed Analysis subtask of the PAN 2012 competition. Our experiments deal with monolingual plagiarism cases, only. We use a simple set-based algorithm, that employs Dice's coefficient as a similarity measure. Furthermore we employ basic strategies from Information Retrieval and Natural Language Processing for stop word removal and language detection. We achieved the 7th place with an overall PlagDet score of 0.35.

1 Introduction

Plagiarism is the misuse of another author's thoughts and ideas without giving credit to the original author. The majority of researchers worldwide deem such plagiarism as an dishonorable act of thievery, that has to be confronted. This confrontation comprises prevention as well as detection of plagiarism, but the manual plagiarism detection is time-consuming and error-prone. Naturally there is a need for automated detection methods.
This paper describes our approach to the Detailed Comparison subtask, that was submitted to the PAN 2012 competition. The algorithm is inspired by a standard model of Information Retrieval: the Bag-of-words model. Therefore natural text is treated as a set of words in the course of this paper.

2 Detailed Comparison Task

2.1 A Set-Based Algorithm

Our detection algorithm is simple and works as follows: in a first step we use a language detection algorithm to determine the natural language the document is written in. This language detector is based on the classification algorithm from [2]. The classifier was trained to distinguish between 11 European languages - we used the Europarl Corpus[3] as our training set. The detector works well most of the time, but there is a considerable problem with documents, which contain text in more than one language. This was the case for a few documents in the PAN 2012 training corpus.
The next processing step deals with the tokenization of the raw text, whereby we store the original positions of the token within the document. We perform only a simple string normalization by collapsing whitespace.

Afterwards the token sequence (for both the suspicious and the source document) has to be divided into smaller sequences. This processing step is called chunking. The selection of a suitable chunk size is crucial for the overall performance of the plagiarism detector: a small chunk size leads usually to more accurate results, but the computation effort rises. A longer chunk size tends to reduce the computation complexity, but the overall precision decreases, because longer chunks contain non-plagiarized text with a higher probability than smaller ones. We use a chunking algorithm with a semi-fixed window size of 250 characters. To prevent tokens from being split, the algorithm expands the window by 1 character at a time until a separator character (usually a single space) is hit. We experimented with another chunking algorithm, which produces chunks on a sentence level (1 natural sentence = 1 chunk), but we drop this approach due to a below average implementation of the sentence detection. So we decided to ”emulate” a sentence detection algorithm by defining the rough length of a sentence with approximately 250 characters.
At this point we chunked the suspicious document into n and the source document into m chunks. We conduct an exhaustive search by comparing all n suspicious chunks to the m source chunks, therefore we have to perform $n \times m$ comparisons by computing the pair-wise similarity of the chunks. Before we compute the similarity of two chunks, we use language-dependent pre-compiled stop lists to eliminate stop words (words with a high frequency are usually considered useless or even harmful for comparison tasks). By removing these words, we prevent our algorithm from detecting similarities between chunks, which are solely based on shared articles, pronouns, etc.
The adjusted chunks are then treated as a simple set of words, so we can use the Dice coefficient to compute the similarity of the sets. Dice’s coefficient is defined as follows:

$$\frac{2 \times |A \bigcap B|}{|A| + |B|}$$

Obviously the function’s range is the interval $[0, 1]$, whereby 0 means no similarity between the sets and 1 indicates total equality. We consider a chunk (or set) as plagiarized, if the Dice similarity between the sets is higher (or equal) 0.4. The Dice similarity is relatively robust against simple obfuscation techniques like re-ordering or word substitution (to a certain degree). Due to the simple approach we are not able to detect cross-language plagiarisms adequately.

2.2 Post-Processing

The described detection algorithm produces a series of detected plagiarism cases, which are analyzed further in a post-processing step. We used non-overlapping chunks (cf. section 2.1), so there is never a plagiarism case contained within another, but adjacent cases are most probable to appear. Therefore two plagiarism cases are merged, if their distance to each other is less than 500 characters.

2.3 Parameter Tuning

The algorithm described in section 2.1 is adjustable by two parameters: the chunk size (in characters) and the similarity threshold. We performed a series of experiments to determine the optimal parameter pair for a high PlagDet score. Due to runtime issues (which remained unsolved at this point), we were unfortunately unable to perform the experiments on the whole PAN 2012 training corpus. Therefore we generated a smaller subcorpus with roughly 10% of the size of the original corpus and performed our experiments on the smaller one. The experimental results are shown in table 1. A chunk

Table 1. Detection Efficiency (showing **C**hunk **S**ize, **S**imilarity **T**hreshold, **P**lag**D**et score, **R**ecall, **P**recision and **G**ranularity)

CS	ST	PD	R	P	G	CS	ST	PD	R	P	G
10000	0.2	0,12	0,29	0,10	1,09	1000	0.2	0,29	0,37	0,57	1,33
	0.4	**0,28**	0,25	0,45	1,00		0.4	**0,40**	0,36	0,78	1,08
	0.6	0,22	0,20	0,33	1,00		0.6	0,24	0,24	0,67	1,19
	0.8	0,17	0,17	0,17	1,00		0.8	0,17	0,17	0,33	1,18
	1.0	0,17	0,17	0,17	1,00		1.0	0,17	0,17	0,17	1,00
5000	0,2	0,20	0,30	0,32	1,08	500	0.2	0,22	0,39	0,47	1,77
	0,4	**0,31**	0,28	0,58	1,02		0.4	**0,38**	0,39	0,96	1,46
	0,6	0,23	0,21	0,33	1,00		0.6	0,22	0,24	0,66	1,38
	0,8	0,17	0,17	0,33	1,00		0.8	0,17	0,17	0,33	1,21
	1,0	0,17	0,17	0,17	1,00		1.0	0,17	0,17	0,17	1,00
2500	0.2	0,27	0,33	0,49	1,12	**250**	0.2	0,14	0,36	0,34	2,20
	0.4	**0,36**	0,33	0,71	1,02		0.4	**0,43**	0,42	0,97	1,27
	0.6	0,23	0,22	0,50	1,04		0.6	0,24	0,27	0,67	1,35
	0.8	0,17	0,17	0,33	1,00		0.8	0,18	0,18	0,33	1,20
	1.0	0,17	0,17	0,17	1,00		1.0	0,17	0,17	0,17	1,00

size of approximately 250 characters with a similarity threshold of 0.4 seemed to be a good choice for our algorithm. We performed no further experiments with even smaller chunk sizes. It is noticeable, that regardless of the chunk size all experiments with a similarity threshold of 0.4 achieved the highest PlagDet score within their respective series.

3 Evaluation

Despite our simple approach we achieved 7th place (as a first-time contestant) in the Detailed Comparison sub-task of the Plagiarism Detection task of the PAN 2012 workshop. Table 2 shows the final results for our contribution. We achieved an overall PlagDet score of 0.35, that was mainly biased by a low recall of 28%. The low recall can be explained by numerous reasons. The most obvious explanation is the simple nature of our detector, because words from the analyzed documents are not subject to further processing like stemming (e.g. [4]). Therefore even minor variations on words (e.g. inflections) are most likely to mislead our detection algorithm. Furthermore it is quite probable, that our detection algorithm fails to detect the exact boundaries of a plagiarism case, because of the used chunking algorithm (cf. section 2.1). A different chunking algorithm, that

produces overlapping chunks of texts, could be a better choice. At last the PAN 2012 training corpus contained some plagiarism cases, which were shorter than our chosen chunk size (cf. section 2.3). If the PAN 2012 test corpus contained such small cases as well, then it is highly probable our detector missed them.
Our algorithm required nearly 28 seconds to analyze a document pair. This runtime issues can easily be explained by a poor implementation of our algorithm. Actually the current version of our detector is much faster and requires only about 10% of the time to analyse a document pair, but naturally the memory requirements increased. We achieved a good precision ($\approx$ 78%) and granularity score (1.27), so there is little requirement on enhancing our merging algorithm at this time.

Table 2. Final Results

PlagDet	Precision	Recall	Granularity	Runtime
0.3499632	0.7762456	0.2820147	1.2692041	27.6457962

4 Conclusion and Future Works

The proposed algorithm is at least suitable to detect monolingual plagiarism cases with low obfuscation, but most probably fails on highly obfuscated plagiarism. Therefore our future work includes the employment of language-dependent stemming techniques (e.g. [4]) to reduce the effect of inflections. Furthermore we plan to employ another chunking algorithm, that is similar to the shingle approach from [1]. Furthermore we plan to extend our approach to deal with cross-language plagiarism cases.

References

1. Broder, A.Z.: Identifying and filtering near-duplicate documents. In: COM '00: Proceedings of the 11th Annual Symposium on Combinatorial Pattern Matching. pp. 1–10. Springer-Verlag (2000)
2. Cavnar, W.B., Trenkle, J.M.: N-gram-based text categorization. In: In Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval. pp. 161–175 (1994)
3. Koehn, P.: Europarl: A Parallel Corpus for Statistical Machine Translation. In: Conference Proceedings: the tenth Machine Translation Summit. pp. 79–86. AAMT (2005)
4. Porter, M.F.: An algorithm for suffix stripping. Program 14(3), 130–137 (1980)