Three way search engine queries with multi-feature document comparison for plagiarism detection

Notebook for PAN at CLEF 2012

Šimon Suchomel, Jan Kasprzak, and Michal Brandejs

Faculty of Informatics, Masaryk University
{suchomel,kas,brandejs}@fi.muni.cz

Abstract In this paper, we describe our approach at the PAN 2012 plagiarism detection competition. Our candidate retrieval system is based on extraction of three different types of web queries with narrowing their execution by skipping certain passages of an input document.
Our detailed comparison system detects common features of input document pair, computing valid intervals from them, and then merging some detections in the postprocessing phase. We also discuss the relevance of current PAN 2012 settings to the real-world plagiarism detection systems.

1 Introduction

Due to the increasing ease of plagiarism the plagiarism detection has nowadays become a need for many institutions. Especially for universities where modern learning methods include e-learning and vast document sources are available online. The core methods for automatic plagiarism detection, which also work in practice on extensive collections of documents, are based on document similarities. In order to compute a similarity we need to possess the original and the plagiarized document.

In the first section we will introduce methods, which took part in PAN 2012 competition[1] in plagiarism detection, for candidate document retrieval from online sources. The task was to retrieve a set of candidate source documents that may had served as an original for plagiarism. During the competition, there were several measures of performance such as: i) number of queries submitted, ii) number of web pages downloaded, iii) precision and recall of web pages downloaded regarding the actual sources, iv) number of queries until the first actual source is found, v) number of downloads until the first actual source is downloaded. Nevertheless, the overall performance measure was not set, thus we mainly focus on minimizing the query workload.

In the second section we describe our approach to detailed document comparison. We highlight the differences of this approach to the one we used for PAN 2010 competition. We then provide the outline of the algorithm, and describe its steps in detail. We briefly mention the approaches we have explored, but did not use in the final submission. Finally, we discuss the performance of our system (both in terms of the plagdet score, and in terms of CPU time).

[1] http://pan.webis.de/

2 Candidate Document Retrieval

The basic concept of candidate document retrieval problem is to use a Web search engine to find suitable documents. In PAN 2012 competition we used ChatNoir search engine [8] which indexes the entire English part of the ClueWeb09[2] corpus. We can now reduce the problem to construct appropriate queries for ChatNoir search engine. The goal is to retrieve similar documents, which contain the same passages as the source document or similar documents dealing with the same theme. The strongest emphasis has been placed on minimizing the number of executed queries since they may be charged, time limited or number limited in the real-world. For instance if we find a similarity for certain part of the document, we do not search for more similarities in that part. We can divide constructed queries into three main groups according to their origin: i) document keywords based, ii) intrinsic plagiarism based, and iii) headers based.

2.1 Keywords Based Queries

The keywords based queries are constructed from automatically extracted keywords from the given document. As for keyword extraction we used methodology based on word repetition similar to method described by Scott [7]. The basic principle claims that the more frequent a single word in a given document is, the more likely it is to be key in it. However the most frequent words usually do not carry any meaning. For example articles (*the*), conjunctions (*as*) or prepositions (*at*). These types of words are usually refered as the stop words. Therefore we firstly filtered out the stop words according to a general English stop words list. Secondly we used a reference corpus word list as an additional filter. We used words in their simple verbatim occurrence allied to a statistical estimate of likelihood. As a result of this, a word to become a key must not exist in the stop list and must occur in the suspicious document relatively much more frequently than in the reference corpus. We used the corpus created from common English Web sources, which contains more then 4 billion tokens [1]. The statistical measure TF-IDF [6] was applied in order to decide whether the processed word shall become a keyword. The TF-IDF combines term frequency (TF) and inverse document frequency (IDF) to produce composite weight of a given term. Taking into account the average length of the suspicious document and minimizing the number of used queries we set the TF-IDF weight threshold to 0.02, which resulted in about 10 to 20 terms identified as keywords per document.

Before executing Web search tasks we must combine extracted keywords into befitting queries. All keyword based queries were constructed from 5 keywords according to their TF-IDF weight consecutively. In other words 5 top ranked keywords combine into the first query, the other 5 top ranked keywords combine into the second query. By using this method we usually obtained from 2 to 3 initial queries per document.

Collocations In addition to this we enhanced subsequent queries by the most frequent collocations of top 4 keywords. For those 4 keywords we extracted the most frequent two-word and three-word collocation occurring within the single suspicious document.

[2] http://boston.lti.cs.cmu.edu/clueweb09/wiki/

A scope for finding collocates was a single sentence. We counted collocates as the most frequent neighbouring words within the scope also omitting stop words. To take advantage of extracted collocations we combined them among selfs also into 5 word queries. After omitting queries containing duplicated words we added, on average, two new queries to each suspicious document. Despite of being composed from collocations we count them also as the keywords based queries. Together with the queries created only from bare keywords, those queries were unconditionally executed as the initial queries.

Queries containing around 5 words should be optimal in order to retrieve highest mean average precision of results. It means that we would like to extract as many results as possible which will still be dealing with the same theme as the source document. Expecting the order of words within a single query has small impact on results, no other heuristics of combining keywords into queries were applied also in order to keep the query amount minimal. In 32 from 33 input test documents there were more than a hundred resulting documents found using the initial queries, which gave a decent document base covering the source document theme for document comparison.

All those previously mentioned queries characterize the document as a whole. They should cover the theme of the document expecting the single document is actually dealing with one theme.

2.2 Intrinsic Plagiarism Based Queries

The intrinsic plagiarism based queries were constructed from a representative sentence residing in the selected part of the document. The representative sentence is any sentence which is greater then eight words in length. Such a sentence should produce the least searching false-positives [5].

The main task concerning representative sentences is to locate the suitable text part. This part should be located by a method leading to discover potential plagiarism inside the document of its own, which is called intrinsic plagiarism detection. Such a method is based on for example text statistics or syntactic features. We chose statistical vocabulary richness method called average word frequency class described in [2]. By using this method we can compute average word frequency class values (AWFC) for arbitrary parts of the text. The change of this value between two adjacent text windows indicates change of the writing style, which can be caused by a plagiarism. We computed AWFC value for every text window of 45 words size, which was shifted through the text by 40 words span. The both values were set empirically during training. Windows with largest change of AWFC compared to AWFC of their neighbouring previous window were selected as a candidate for query extraction.

The query from the representative sentence was composed by 6 words from the beginning of the sentence, also omitting stop words. We chose 6 words to make more specific queries and not to exceed the phrasal ChatNoir query limit. Unfortunately in the time of PAN 2012 competition the ChatNoir search engine did not support phrasal search. Since querying a single sentence is a phrasal search, one might have expected to achieve even better results from those queries.

The intrinsic plagiarism based queries are document positionable, meaning that we can store their position within the document. It is due to the fact that those queries are

created from more or less subsequent words. As a result of that, they are executed conditionally, for more information see Section 2.4. They characterize only a specific portion of the suspicious document on the contrary to keyword based queries. The positions within the document were set to the start positions of the representative sentences.

2.3 Headers Based Queries

The last type of queries were constructed from local headers of the document. A header usually characterizes briefly and accurately the following section, thus it can be used as a search query. In real scenarios the headers should be located using some document metadata. In that way we can distinguish for example more header levels. In the case of PAN 2012 competition we constructed headers based queries according to several basic rules: i) the query is created from any line which has an empty line above and bellow, ii) it is from 2 to 6 words in length also omitting stop words. These basic rules applied on most headers from the given format of test corpus text files.

Note that the length of headers based queries are variable, thus they can be both more specific and more general.They are also document positionable. We calculated their position as the start position of the following text.

2.4 Combination and Execution of Queries

All queries from a given document were firstly preconstructed and they were afterwards sequentially executed according to their priority: all keywords based, intrinsic plagiarism based and headers based. During the process we could omit some of the positionable queries, in order to lower total number of executed queries. The condition for their omitting is described further in this section.

After each executed query the intermediate results of plagiarism were calculated using the multi feature document comparison techniques described in Section 3. The queries processing is outlined as Algorithm 1, where for simplicity snippet calculation and Web source download is omitted. After executing a query the intermediate results were always calculated as follows:

For every query result in a result page based on a snippet to input suspicious document similarities, we decided whether to actually download the resulting URL or not. Since the snippet contains parts of the Web document to each given keyword, we calculated pre-similarity as apportionment of every word match between the snippet and the suspicious document. If there were at least 80% match, we downloaded the Web source for thorough investigation.

For each downloaded Web document we calculated similarities with the input suspicious document as a document pair described in Section 3. All similarities were stored in form of intervals from within the input suspicious document. In other words for every similar part between the downloaded document and the suspicious document we stored the beginning position and the ending position of that part in the suspicious document.

As a result of that, during processing further queries which haven't been executed yet, we can omit those of which document position intersects with any of already found similarities intervals. All downloaded documents, which have at least one similarity interval, are declared as possible source of a plagiarism.

Algorithm 1 Queries execution

Input: suspicious document d, queries $Q_{KW}, Q_{Intrinsic}, Q_{Headers}$
Output: plagiarism source candidate Web documents W_{plag}

$I \leftarrow \emptyset; I \in \mathbb{N}^2$
$W \leftarrow \emptyset$
for all $q \in (Q_{KW} \cup Q_{Intrinsic} \cup Q_{Headers})$ **do**
    if $position(q)$ does not intersect with any interval $\boldsymbol{i} \in I$ **then**
        $W \leftarrow execute(q)$
        for all $w \in W$ **do**
            $I_{sub} \leftarrow get\_similarities(w, d)$
            $I \leftarrow I \cup I_{sub}$
            if $I_{sub} \neq \emptyset$ **then**
                $W_{plag} \leftarrow W_{plag} \cup \{w\}$
            end if
        end for
    end if
end for
return W_{plag}

2.5 Queries Comparison

In total there were extracted 133 keywords based queries, 165 intrinsic plagiarism based queries and 331 headers based queries from the test corpus during the test phase. Table 1 compares results according to query type.

Query type	Extracted	Omitted	Similarities portion
KW	4.16	N/A	72.5%
Intrinsic	5.16	2.35	24.3%
Headers	10.34	4.75	3.2%

Table 1. Queries type comparison.

The second column represents the arithmetic mean of the query count per document. The third column represents the arithmetic mean of the omitted query count per document. The fourth column shows total portion of similarities found, taking into account the number of similarities regardless of interval sizes. We can see that nearly half of the prepared queries were omitted due to the fact, that there had been found a similarity covering their document position. We can also see that there were detected about 5 cases of potential plagiarism on average, by means of used AWFC intrinsic plagiarism detection method. Table 1 also shows keywords based queries as the most successful and headers based queries as the least successful. Despite the fact, that they were most numerous they ended with only more than a 3% of total similarities found. Nevertheless, please note that the headers based queries were executed as the last, thus they were

used only for finding undiscovered potential similarities. In order to really compare the query type performance, we would need to execute and evaluate them separately.

To conclude this section we can say, that all types of queries were successful. The headers based were executed as the last and in the process they ranked in the lowest performance. The interesting finding is the fact, that we can even greatly lower the number of executed queries. By omitting all of headers based queries we could lower the total number of executed queries by 45% with only 3.2% of recall lost.

	Total Workload		Time to 1st Result		No	Reported Srcs.		Downloaded Srcs.		Retrieved Srcs.	
Team	**Queries**	**Downloads**	**Queries**	**Downloads**	**result**	**Prec.**	**Recall**	**Prec.**	**Recall**	**Prec.**	**Recall**
Gillam et al. University of Surrey, UK	63.44	527.41	4.47	25.88	**1**	0.6266	0.2493	0.0182	**0.5567**	0.0182	**0.5567**
Jayapal University of Sheffield, UK	67.06	173.47	8.78	13.50	1	**0.6582**	**0.2775**	0.0709	0.4342	**0.0698**	0.4342
Kong Leilei Heilongjiang Institute of Technology, China	551.06	326.66	80.59	27.47	2	0.5720	0.2351	0.0178	0.3742	0.0141	0.3788
Palkovskii et al. Zhytomyr State University, Ukraine	63.13	1026.72	27.28	318.94	6	0.4349	0.1203	0.0025	0.2133	0.0024	0.2133
our approach	**12.56**	**95.41**	**1.53**	**6.28**	2	0.5177	0.2087	**0.0813**	0.3513	0.0094	0.4519

Table 2. PAN 2012 candidate document retrieval results.

Table 2 shows results of PAN 2012 candidate document retrieval task as averages over the all 32 documents from the test corpus. Our approach led to obtain decent retrieval performance with the minimal total workload and minimal time to 1st result. Also the 80% word match threshold in a Web snippet appears to be suitable, since we also achieved the highest precision among downloaded sources.

3 Detailed Document Comparison

The detailed comparison task of PAN 2012 consisted in a comparison of given document pairs, with the expected output being the annotation of similarities found between these documents. The submitted program was running in a controlled environment separately for each document pair, without the possibility of keeping any cached data among runs.

3.1 Differences Against PAN 2010

Our approach in this task is loosely based on the approach we used in PAN 2010 [3]. The main difference is that instead of looking for similarities of one type (for PAN 2010, we have used word 5-grams), we developed a method of evaluating multiple types of similarities (we call them *common features*) of different properties, such as density and length.

As a proof of concept, we used two types of common features: word 5-grams and stop word 8-grams, the later being based on the method described in [10].

In addition to the above, we made several minor improvements to the algorithm such as parameter tuning and improving the detections merging in the post-processing stage.

3.2 Algorithm Overview

The algorithm evaluates the document pair in several stages:

- tokenizing both the suspicious and source documents
- forming *features* from some tokens
- discovering *common features*
- making *valid intervals* from common features
- postprocessing

3.3 Tokenization

We tokenize the document into words, where word is a sequence of one or more characters of the *Letter* Unicode class. With each word, two additional attributes needed for further processing, are associated: the offset where the word begins, and the word length.

The offset where the word begins is not necessarily the first letter character of the word itself. We discovered that in the training corpus some plagiarized passages were annotated including the preceding non-letter characters. We used the following heuristics to add parts of the inter-word gap to the previous or the next adjacent word:

- When the inter-word gap contains interpunction (any of the dot, semicolon, colon, comma, exclamation mark, question mark, or quotes):

    - add the characters up to and including the interpunction character to the previous word,
    - ignore the space character(s) after the interpunction character,
    - add the rest to the next word.
- Otherwise, when the inter-word gap contains newline:
    - add the character before the first newline to the previous word,
    - ignore the first newline character,
    - add the rest to the next word.
- Otherwise: ignore the inter-word gap characters altogether.

When the detection program was given three different files instead of two (meaning the third one is machine-translated version of the second one), we tokenized the translated document instead of the source one. We used the line-by-line alignment of the source and machine-translated documents to transform the word offsets and lengths in the translated document to the terms of the source document.

3.4 Features

We have used features of two types:

- Lexicographically sorted word 5-grams formed of words at least three characters long.
- Unsorted stop word 8-grams formed from 50 most frequent words in English, as described in [10]. We have further ignored the 8-grams, formed solely from the six most frequent English words (*the*, *of*, *and*, *a*, *in*, *to*), or the possessive *'s*.

We represented each feature with the 32 highest-order bits of its MD5 digest. This is only a performance optimization targeted for larger systems. The number of features in a document pair is by several orders of magnitude lower than 2^{32}, thus the probability of hash function collision is low. For pair-wise comparison, it would be feasible to compare the features directly instead of their MD5 sums.

Each feature has also two attributes: offset and length. Offset is taken as the offset of the first word in a given feature, and length is the offset of the last character in a given feature minus the offset of the feature itself.

3.5 Common Features

For further processing, we took into account only the features present both in source and suspicious document. For each such *common feature*, we created the list of $(\text{offset}, \text{length})$ pairs for the source document, and a similar list for the suspicious document. Note that a given feature can occur multiple times both in source document and suspicious document.

3.6 Valid Intervals

To detect a plagiarized passage we need to find a set of common features, which map to a dense-enough interval both in the source and suspicious document. In our previous work we described the algorithm for discovering these *valid intervals* [4]. A similar approach is also used in [10]. Both of these algorithms use features of a single type, which allows to use the ordering of features as a measure of distance.

When we use features of different types, there is no natural ordering of them. For example a stop word 8-gram can span multiple sentences, which can contain several word 5-grams. The assumption of both of the above algorithms, that the last character of the previous feature is before the last character of the current feature, is broken.

We modified the algorithm for computing valid intervals with multi-feature detection to use character offsets only instead of feature order numbers. We used valid intervals consisting of at least 4 common features, with the maximum allowed gap inside the interval (characters not belonging to any common feature of a given valid interval) set to 4000 characters.

3.7 Postprocessing

In the postprocessing phase we took the resulting valid intervals and made attempt to further improve the results. We firstly removed overlaps. If both overlapping intervals were shorter than 300 characters, we have removed both of them. Otherwise, we kept the longer detection (longer in terms of length in the suspicious document).

We then joined the adjacent valid intervals into one detection, if at least one of the following criteria were met:

- the gap between the intervals contained at least 4 common features, and it contained at least one feature per 10,000 characters[3]
- the gap was smaller than 30,000 characters and the size of the adjacent valid intervals was at least twice as big as the gap between them
- the gap was smaller than 30,000 characters and the number of common features per character in the adjacent interval was not more than three times bigger than number of features per character in the possible joined interval.

3.8 Results

These parameters were fine-tuned to achieve the best results on the training corpus. With these parameters our algorithm got the total plagdet score of 0.7288 on the training corpus. The details of the performance of our algorithm are presented in Table 3. In the PAN 2012 competition, we have acchieved the plagdet score of 0.6827, precision of 0.8932, recall of 0.5525, and granularity of 1.0000.

[3] we have computed the length of the gap as the number of characters between the detections in the source document, plus the number of charaters between the detections in the suspicious document.

	plagdet	recall	precision	granularity
whole corpus	0.7288	0.5994	0.9306	1.0007
01-no-plagiarism	0.0000	0.0000	0.0000	1.0000
02-no-obfuscation	0.9476	0.9627	0.9330	1.0000
03-artificial-low	0.8726	0.8099	0.9477	1.0013
04-artificial-high	0.3649	0.2255	0.9562	1.0000
05-translation	0.7610	0.6662	0.8884	1.0008
06-simulated-paraphrase	0.5972	0.4369	0.9433	1.0000

Table 3. Performance on the training corpus

3.9 Other Approaches Explored

There are several other approaches we evaluated, but which were omitted from our final submission for various reasons. We think mentioning them here is worthwhile nevertheless:

Intrinsic Plagiarism Detection We tested the approach based on character n-gram profiles of the interval of the fixed size (in terms of n-grams) and their differences to the profile of the whole document [11]. We have further enhanced the approach with using gaussian smoothing of the style-change function [3]. For PAN 2012 we made further improvements to the algorithm, resulting in more stable style change function in both short and long documents.

We tried to use the results of the intrinsic plagiarism detection as hint for the post-processing phase, allowing to merge larger intervals, if they both belong to the same passage detected by the intrinsic detector. This approach did not provide improvement when compared to the static gap limits, as described in Section 3.7, therefore we have omitted it from our final submission.

Cross-lingual Plagiarism Detection For cross-lingual plagiarism detection, our aim was to use the public interface to Google Translate[4] if possible, and use the resulting document as the source for standard intra-lingual detector. Should the translation service not be available, we wanted to use the fall-back strategy of translating isolated words only, with the additional exact matching of longer words (we have used words with 5 characters or longer). We have supposed that these longer words can be names or specialized terms, present in both languages.

We used dictionaries from several sources, for example *dicts.info*[5], *omegawiki*[6], and *wiktionary*[7].

In the final submission, we simply used the machine translated texts, which were provided to the running program from the surrounding environment.

[4] http://translate.google.com/
[5] http://www.dicts.info/
[6] http://www.omegawiki.org/
[7] http://en.wiktionary.org/

3.10 Further discussion

From our previous PAN submissions, we knew that the precision of our system was good, and because of the way how the final score is computed, we wanted to exchange a bit worse precision for better recall and granularity. So we pushed the parameters towards detecting more plagiarized passages, even when the number of common features was not especially high.

Plagdet score Our results from tuning the parameters show that the plagdet score [9] is not a good measure for comparing the plagiarism detection systems. For example, the gap of 30,000 characters, described in Section 3.7, can easily mean several pages of text. And still the system with this parameter set so high resulted in better plagdet score.

Another problem of plagdet can be seen in the 01-no-plagiarism part of the training corpus. The border between the perfect score 1 and the score 0 is a single false-positive detection. Plagdet does not distinguish between the system reporting this single false-positive and the system reporting the whole data as plagiarized. Both get the score 0. However, our experience from real-world plagiarism detection systems shows that the plagiarized documents are in a clear minority, so the performance of the detection system on non-plagiarized documents is very important.

Performance Notes We consider comparing the CPU time performance of PAN 2012 submissions almost meaningless, because any sane system would precompute features for all documents in a given set of suspicious and source documents, and use the results for pair-wise comparison, expecting that any document will take part in more than one pair.

Also, the pair-wise comparison without caching any intermediate results leads to worse overall performance. In our PAN 2010 submission one of the post-processing steps was to remove all the overlapping detections from a given suspicious documents, when these detections were from different source doucments and were short enough. This removed many false-positives and improved the precision of our system. This kind of heuristics was not possible in PAN 2012.

As for the performance of our system, we split the task into two parts: i) finding the common features, and ii). computing valid intervals and postprocessing. The first part is more CPU intensive and the results can be cached. The second part is fast enough to allow us to evaluate many combinations of parameters.

We did our development on a machine with four six-core AMD 8139 CPUs (2800 MHz), and 128 GB RAM. The first phase took about 2500 seconds on this host, and the second phase took 14 seconds. Computing the plagdet score using the official script in Python took between 120 and 180 seconds, as there is no parallelism in this script.

When we tried to use intrinsic plagiarism detection and language detection, the first phase took about 12,500 seconds. Thus omitting these features clearly provided huge performance improvement.

Our code was written in Perl and had about 669 lines, not counting comments and blank lines.

4 Conclusions

We present methods for candidate document retrieval which were also tested during the PAN 2012 competition. The proposed methods are applicable in general to any type of text input with no apriori information about the input document. In PAN 2012 competition the proposed methods succeeded with competitive amount of plagiarism detected with only a small fraction of used queries compared to the others.

We also present a novel approach for detailed (pair-wise) document comparison, where we allow the common features of different types to be evaluated together into valid intervals, even though the particular types of common features can vary to the great extent in their length and importance, and do not provide a natural ordering. The presented approach achieved a second-highest plagdet score in the PAN 2012 competition.

References

1. Sketch Engine EnTenTen Corpus. http://trac.sketchengine.co.uk/wiki/Corpora/enTenTen (2012)
2. Eissen, S.M.Z., Stein, B.: Intrinsic plagiarism detection. In: Proceedings of the European Conference on Information Retrieval (ECIR-06) (2006)
3. Kasprzak, J., Brandejs, M.: Improving the reliability of the plagiarism detection system. In: Notebook Papers of CLEF 2010 LABs and Workshops. Citeseer (2010)
4. Kasprzak, J., Brandejs, M., Křipač, M.: Finding plagiarism by evaluating document similarities. In: SEPLN'09: The 25th edition of the Annual Conference of the Spanish Society for Natural Language Processing (2009)
5. Knight, A., Almeroth, K., Bimber, B.: An automated system for plagiarism detection using the internet. In: Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications, pp. 3619-3625 (2004)
6. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press, Cambridge, UK (2008)
7. Mike Scott and Christopher Tribble: Textual Patterns, Key Words and Corpus Analysis in Language Education. John Benjamins Publishing Company (2006)
8. 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) (Aug 2012)
9. Potthast, M., Stein, B., Barrón-Cedeño, A., Rosso, P.: An Evaluation Framework for Plagiarism Detection. In: Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010). Association for Computational Linguistics, Beijing, China (Aug 2010)
10. Stamatatos, E.: Plagiarism detection using stopword n-grams. Journal of the American Society for Information Science and Technology (2011)
11. Stamatatos, E.: Intrinsic plagiarism detection using character n-gram profiles. In: Proceedings of the SEPLN'09 Workshop on Uncovering Plagiarism, Authorship and Social Software Misuse. pp. 38–46 (2009)