FASTDOCODE: Finding Approximated Segments of N-Grams for Document Copy Detection

Lab Report for PAN at CLEF 2010

Gabriel Oberreuter, Gaston L'Huillier, Sebastián A. Ríos, and Juan D. Velásquez

University of Chile, Department of Industrial Engineering, Santiago, Chile
goberreu@ing.uchile.cl, {glhuilli|srios|jvelasqu}@dii.uchile.cl

Abstract Nowadays, plagiarism has been presented as one of the main distresses that the information technology revolution has lead into our society for which using pattern matching algorithms and intelligent data analysis approaches, these practices could be identified. Furthermore, a fast document copy detection algorithm could be used in large scale applications for plagiarism detection in academia, scientific research, patents, knowledge management, among others. Notwithstanding the fact that plagiarism detection has been tackled by exhaustive comparison of source and suspicious documents, approximated algorithms could lead to interesting results. In this paper, an approach for plagiarism detection is presented. Results in a learning dataset of plagiarized documents from the PAN'09, and its further evaluation in the PAN'10 plagiarism detection challenge, showed that the trade-off between speed and performance could improve other plagiarism detection algorithms.

1 Introduction

Plagiarism in academia is rising and multiple authors have worked to describe this phenomena [11, 12, 20]. As commented by Hunt in [11], "Internet Plagiarism" is referred sometimes as a cataclysmic consequence of the "Information Technology revolution", as it proves to be a big problem in academia. In [20], plagiarism is analyzed from various perspectives and considered as a problem that is growing bigger over time. In [12], the author analyzes different statistical data and the implications of the "IT age".

To tackle this problem, one approach is to try to detect plagiarism. Different methods involving computer aided plagiarism detection have been under research [6, 8, 10, 14, 16, 24, 25], from which different system for automatic plagiarism detection have been developed. However, different ways for neutralizing such detection systems have been presented. Such methods usually involve modifying the text in such way that the presentation of the document remains the same, but the underlying code is different and normally this differentiation render the detection systems useless [19].

Plagiarism detection for document sources can be classified into several categories [7]. From exact document copy, to paraphrasing, different levels of plagiarism techniques can been used in several contexts [14, 28]. Likewise, pairs of documents can be described into different categories as unrelated, related, partly overlapped, subset, and copied.

When comparing a suspicious document against a collection of possible sources, it is tried to identify the sentences, paragraph or ideas that have been copied. This is called external plagiarism detection [22], and multiple efforts are being oriented in this area. Another approach, is to determine within features extracted from just one given document. However, this work is mainly focused on external plagiarism detection, without considering elements from the intrinsic plagiarism detection case.

The main contribution of this work is a technique for plagiarism detection based on a two step evaluation. First, a filter evaluation which considers a fast generation of segments of n-grams for an approximated decision. And second, an obfuscation and exhaustive search process for the offset and length of the plagiarized extraction between two previously classified documents is performed. This two-step algorithm is based on different document pre-processing strategies and decision thresholds which gives a large number of parameters or degrees of freedom to be determined.

This paper is structured as follows: In Section 2 an overview of plagiarism detection algorithms and related work is presented. Then, in Section 3 the proposed FAST Document Copy Detection (FASTDOCODE) method is introduced. Afterwards, in Section 4, the experimental setup and evaluation performance criteria are described. In Section 5 results are discussed. Finally, in Section 6 the main conclusions are presented.

2 Related Work

According to Schleimer et al. [23], copy prevention and detection methods can be combined to reduce plagiarism. While copy detection methods can only minimize it, prevention methods can fully eliminate it and decrease it. Notwithstanding this fact, prevention methods need the whole society to take part, thus its solution is non trivial. Copy or plagiarism detection methods tackle different levels, from simple manual comparison to complex automatic algorithms [22]. Among these techniques, document similarity detection, writing style detection, document content similarity, content translation, multi-site source plagiarism, and multi-lingual plagiarism detection methods have been previously proposed [6, 7, 14, 18, 22, 23, 26].

2.1 Intrinsic Plagiarism Detection

When comparing texts against a reference set of possible sources, comes the complication of choosing the right set. And now more than ever, with the possibilities the internet bring to plagiarists, this task becomes more complicated to achieve. For this, intrinsic plagiarism detection algorithms have been developed [28].

The writer style can be analyzed within the document and an examination for incongruities can be done. The complexity and style of each text can be analyzed based on certain parameters such as text statistics, syntactic features, part-of-speech features, closed-class word sets, and structural features [28]. Whose main idea is to define a criterium to determine if the style has changed enough to indicate plagiarism.

It is important to note that using intrinsic plagiarism detection for both, automated and manual, it is not demonstrated that a paragraph or a part of the document is being copied, because there is no reference to compare to. Therefore this kind of plagiarism detection category is only indicative and should be used in conjunction with human supervision. Nevertheless, intrinsic plagiarism is useful when trying to discover originality and authorship of a document.

2.2 External Plagiarism

Before the comparison between each possible source and the suspicious document can be executed, an important obstacle is to be resolved. This task consist in defining and gathering the possible sources, and this is becoming more and more complex as the technology becomes more available. In [5], the suspicious document is chopped into queries and web search engines are used to obtain a set of candidates sources. This approach helps tackle this problem but more work is needed.

Another issue to be considered, is when the collection of possible sources become too large. The size of the set of possible sources can be thousands of documents. In [22], PAN Competition and Workshop, the external part of the competition, and now merged with the training corpus, considers a set of sources of 14,428 documents or possible sources. In this case solutions do exist, as reducing the search space using different data mining techniques.

In [17], the use of n-grams for plagiarism detection is explored. The use of n-grams gives some flexibility to the detection, as reworded fragments could still be detected. In particular, in [3] the tri-gram structure is found to be the most effective in this task. This method is possible because the common n-grams between two documents are usually a low percentage of the total number of n-grams of both text. Due to this, n-grams could probe promissory for plagiarism detection techniques. Furthremore, in [16], Lyon et al. extended their work and the Ferret system was implemented, which uses this approximation to detect plagiarism. A distance is calculated between the documents, based on the n-grams found in common. The results indicate that this structure is useful and provides flexibility at detecting plagiarism with modifications of words.

Other approaches focus on solving the plagiarism detection problem as a traditional classification problem from the machine learning community [1, 9, 13]. Bao et al. in [13] and then in [1], proposed to use a Semantic Sequence Kernel (SSK), and then using it into a traditional Support Vector Machines (SVMs) formulation based on the Structural Risk Minimization (SRM) [4, 27] principle from statistical learning theory, where the general objective is finding out the optimal classification hyperplane for the binary classification problem (plagiarized, not plagiarized). Likewise, other approaches solves the same classification problem by using Self Organizing Feature Maps (SOFM) [15], with promising results in the classification performance.

An interesting issue is the multi-lingual and cross-language detection of plagiarism. This topic is currently under research [2, 21], where promising results for plagiarism and cross-lingual information retrieval have been presented.

2.3 Reducing Search Space

One of the issues to be resolved in external plagiarism analysis and detection is the number of source document candidates. When the task is to detect plagiarism between a small set of suspicious against a small set of source documents, it is simple to search for plagiarism in every pair of documents. The problem is presented when the universe of possible sources is not well defined, or the set of documents is too large. In this case the approach need to be modified, and those changes usually consists in adding a step in the process of plagiarism discovery: the search space reduction.

The aim of this step is to effectively and efficiently identify which texts are possible sources of plagiarism, if any. Usually multiple statistical tools are used in order to reduce the computational time required for computing a large corpus of documents while trying to maintain accuracy at determining which sources need to be discarded.

3 Proposed Method

In this section, the main contribution of our work is described. In the first place, the overall FASTDOCODE algorithm is presented in terms of previously introduced notation. Then, the two steps that defines FASTDOCODE, that is, the approximated segment finding algorithm and the exhaustive offset and length search, are presented in subsections 3.2 and 3.3 respectively. In this section all algorithms are presented as pseudocode, together with a brief description on how different parameters could be used.

Let us introduce some concepts. In the following, let $\mathcal{V}$ a vector of words that defines the vocabulary to be used. We will refer to a word w, as a basic unit of discrete data, indexed by $\{1, ..., |\mathcal{V}|\}$. A document d is a sequence of S words ($|d| = S$) defined by $\mathbf{w} = (w^1, ..., w^S)$, where w^s represents the s^{th} word in the message. Finally, a corpus is defined by a collection of $\mathcal{D}$ documents denoted by $\mathcal{C} = (\mathbf{w}_1, ..., \mathbf{w}_{|\mathcal{D}|})$.

3.1 FASTDOCODE

Given a wide set of parameters $\mathcal{D}_{\text{source}}, \mathcal{D}_{\text{suspicious}}, n, k, m,$ SORTSTRATEGY$, \theta_1, \theta_2, \tau_{min}, St, Pe$, the algorithm tries to find for a corpus $\mathcal{C} = \{\mathcal{D}_{\text{source}}, \mathcal{D}_{\text{suspicious}}\}$ all plagiarized documents in the suspicious partition, using as search space the source partition. This algorithm is based on external plagiarism detection, and does not include intrinsic plagiarism nor multi-lingual evaluation. In general terms, the algorithm first reduces the search space by using an approximated search of segments of n-grams, and then within selected pairs of documents, using an exhaustive search algorithm, finds the offset and its length for both exact and obfuscated copy.

Algorithm 3.1: FAST-DOCODE

Data: $\mathcal{D}_{\text{source}}, \mathcal{D}_{\text{suspicious}}, n, k, m,$ SORTSTRATEGY$, \theta_1, \theta_2, \tau_{min}, St, Pe$
Result: Vector OL with all Offsets and their lengths for the complete corpus of documents

Initialize Vector $pair \leftarrow \{\}$ and $Offset Lenght \leftarrow \{\}$;
foreach $\mathbf{d}_i \in \mathcal{D}_{suspicious}$ **do**
  $(\kappa_i, \mathbf{t}_i) \leftarrow$ PREPROCESSDOCUMENT$(\mathbf{d}_i, n = 3, k, m)$;
  foreach $\mathbf{d}_j \in \mathcal{D}_{source}$ **do**
    $(\kappa_j, \mathbf{t}_j) \leftarrow$ PREPROCESSDOCUMENT$(\mathbf{d}_j, n = 3, k, m)$;
    if APPROXIMATECOMPARISON$(\kappa_i, \kappa_j, \mathbf{t}_i, \mathbf{t}_j, \theta_1, \theta_2)$ **then**
      $p(i, j) \leftarrow (\mathbf{d}_i, \mathbf{d}_j)$;
      $pair$.add(p);
foreach $p \in pair$ **do**
  $\mathbf{t}_i \leftarrow$ PREPROCESSDOCUMENT$(p.\mathbf{d}_i, n = 2, k, m,$ SORTSTRATEGY$)$;
  $\mathbf{t}_j \leftarrow$ PREPROCESSDOCUMENT$(p.\mathbf{d}_j, n = 2, k, m,$ SORTSTRATEGY$)$;
  OL.add(FINDOFFSETLENGHT1$(\mathbf{t}_i, \mathbf{t}_j, \tau_{min}, St, Pe)$);
  OL.add(FINDOFFSETLENGHT2$(\mathbf{t}_i, \mathbf{t}_j, \tau_{min}, St, Pe)$);
return OL ;

In algorithm 3.1, the general evaluation of a corpus is presented. In particular, different procedures are used within the code which helps in the preprocessing of documents. The method PREPROCESSDOCUMENT, presented in algorithm 3.2, takes as input a given document, and returns a set of n-grams or segments of n-grams given the case. If $n = 2$, only a set of bi-grams will be computed, and if $n = 3$ a process of finding segments of n-grams will be performed. Segments of n-grams will be intensively used in the approximated search for reducing the search space, whether the bi-grams will be used in finding all offsets and their lengths.

Algorithm 3.2: PREPROCESSDOCUMENT

Data: $\mathbf{d}_i, n, k, m$
if $n = 3$ **then**
  REMOVESTOPWORDS($\mathbf{d}_i$);
  $\mathbf{t}_i \leftarrow$ GENERATENGRAMS($\mathbf{d}_i, n$) ;
  $\mathbf{k}_i \leftarrow$ GENERATEKNGRAMS($\mathbf{t}_i, k$);
  $\mathbf{k}_i^* \leftarrow$ SORT($\mathbf{k}_i$, SORTSTRATEGY) ;
  $\kappa_i \leftarrow$ SELECTMLASTNGRAMS($\mathbf{k}_i^*, m$) ;
  return $(\kappa_i, \mathbf{t}_i)$;
else
  $\mathbf{t}_i \leftarrow$ GENERATENGRAMS($\mathbf{d}_i, n$) ;
  return $\mathbf{t}_i$;

As presented in algorithm 3.2, new methods are introduced for the processing, such as the GENERATENGRAMS function that takes a given document d_i and returns a set of n-grams with the structure $(w_i, w_{i+1} \ldots, w_{i+n}), \forall i \geq 1, n \leq S$. Function GENERATEKNGRAMS, generates groups of length k using all n-grams. Then, a SORT algorithm is used within segments, with a specific sorting strategy. In this research, an alphabet sorting strategy and a Term Frequency sorting strategy where used as a variation on the proposed algorithm. Finally, a SELECTMLASTNGRAMS function, as specified in its name definition, selects only the last m n-grams within the segment. This approach can be considered as an analogy to a sampling strategy for each segment, thus contributing to minimize the number of comparisons to be executed and enhancing the runtime of the algorithm.

3.2 Finding Segments Approximation

Algorithm 3.3: APPROXIMATECOMPARISON

Data: $\kappa_i, \kappa_j, \mathbf{t}_i, \mathbf{t}_j, \theta_1, \theta_2$
if SMATCH($\mathbf{t}_i, \mathbf{t}_j, s \geq 1$) **then**
  if SMATCH($\kappa_i, \kappa_j, s \geq \theta_1$) **then**
    if SMATCH($\mathbf{t}_i, \mathbf{t}_j, s \geq \theta_2$) **then**
      return true ;
    end
  end
end
else
  return false;
end

Once documents d_i and d_j are processed in n-grams and segments of n-grams, t_i, t_j and κ_i, κ_j respectively, a set of conditions are evaluated in order to set the relation that document d_i has with document d_j, that is, if they are somehow related (algorithm 3.3 returns true), or if it is not worthy to keep finding further relationships (algorithm 3.3 returns false). In this sense, this is an approximated finding procedure that considers both n-grams and their k segments to decide if there is enough information to classify as plagiarism or not.

Algorithm 3.3 first evaluates an SMATCH($\mathbf{t}_i, \mathbf{t}_j, s \geq 1$) algorithm which returns true whether at least one n-gram from $\mathbf{t}_i$ matches one n-gram from $\mathbf{t}_j$. Also a variation of previous matching method is used within the segments of n-grams. Condition SMATCH($\kappa_i, \kappa_j, s \geq \theta_1$) states that at least θ_1 n-grams must

match in between segments κ_i and κ_j. If this is hold, the next condition $\text{SMATCH}(\mathbf{t}_i, \mathbf{t}_j, s \geq \theta_2)$ is associated to find whether at least θ_2 n-grams matches between $\mathbf{t}_i$ and $\mathbf{t}_j$. In general terms, this procedure helps on reducing the search space, and improving the algorithm in both execution time and hardware requirements. By using these constraints, it is possible now to go into a further algorithm for finding the needed offset and its length.

3.3 Find the Offset and its Length

Algorithm 3.4 describes roughly how to find the offset and its length within two documents.

Algorithm 3.4: FINDOFFSETLENGTH

Data: $\mathbf{d}_i, \mathbf{d}_j, \mathbf{t}_i, \mathbf{t}_j, \tau_{min}, St, Pe$
// Find obfuscated and textual copy within documents d_i and d_j
foreach $w_i \in \mathbf{d}_i$ **do**
  foreach $w_j \in \mathbf{d}_j$ **do**
    Initialize Vector $bp(i)^{\text{left}} \leftarrow \{\}$, $bp(j)^{\text{right}} \leftarrow \{\}$, $bp(j)^{\text{left}} \leftarrow \{\}$, $bp(i)^{\text{right}} \leftarrow \{\}$;
    // Move the XMATCH in both $\leftarrow$ and $\rightarrow$ sides of the document in steps St and checking that Pe percentage of similar words within the step
    repeat
      $bp(i)^{\text{right}}.add(t_i)$ and $bp(j)^{\text{right}}.add(t_j)$
    until $!\text{XMATCH}(t_i, t_j, s = 1, \leftarrow, St, Pe)$;
    repeat
      $bp(i)^{\text{left}}.add(t_i)$ and $bp(j)^{\text{left}}.add(t_j)$
    until $!\text{XMATCH}(t_i, t_j, s = 1, \rightarrow, St, Pe)$;
    if $max\{|bp(i)^{left} - bp(i)^{right}|, |bp(j)^{left} - bp(j)^{right}|\} > \tau_{min}$ **then**
      $OffsetLength(i)$.add$(bp(i)^{\text{left}}, |bp(i)^{\text{left}} - bp(i)^{\text{right}}|)$;
      $OffsetLength(j)$.add$(bp(j)^{\text{left}}, |bp(j)^{\text{left}} - bp(j)^{\text{right}}|)$;
    end
    // Remove all words inside break points for both d_i and d_j
    REMOVEINCLUDEDWORDS$(\mathbf{t}_i, bp(i)^{\text{left}}, bp(i)^{\text{right}})$;
    REMOVEINCLUDEDWORDS$(\mathbf{t}_j, bp(j)^{\text{left}}, bp(j)^{\text{right}})$;
    UPDATE$(\mathbf{d}_i)$;
    UPDATE$(\mathbf{d}_j)$;
  end
end
return $Offsetlenght$;

Previous algorithm 3.4, finds obfuscated and textual copy within documents d_i and d_j, then a match strategy is moved both left and right side of the document, adding to the offset array the matching segments. Finally, to avoid the search over detected plagiarism passages, the break points are saved and used to remove them.

4 Experiments

In this section, the experimental setup and the evaluation criteria is presented. First, the selected partition of a plagiarism detection corpus from the PAN'09 [22] is discussed together with some of the parameters selected to evaluate different benchmark plagiarism detection algorithms. Then, the evaluation criteria and performance measures used for the training step of the algorithm are presented.

4.1 Experimental Setup

The PAN'09 plagiarism detection corpus [22] was used as a seed to train different plagiarism detection algorithms.

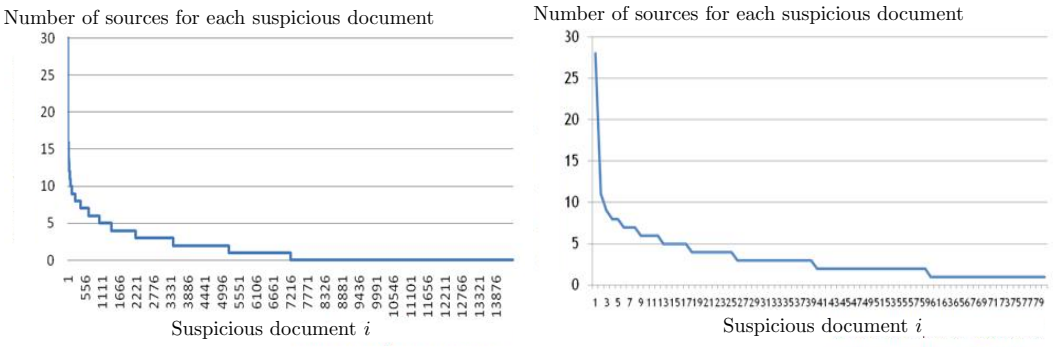

Figure 1. Dataset distribution in terms of the construction of the number of source documents that each suspicious document was originated from.

For the experiment, a small sample of the PAN'09 corpus is chosen. This sample considers only external English plagiarism cases. It is constructed as Figure 1 suggests: maintaining the number of references per suspicious document from the original corpus.

Four algorithms are used for experimental and testing purposes. Three of the selected algorithms are based on the previous approach presented in section 3 and a variation of the `unix diff` command used to detect changes between two documents was used as benchmark. Due to the lack of space, only a brief description of each of the selected algorithms is presented. Further information was intentionally discarded by authors.

The first, named "SimParalell" is an iteration where the pair of documents is compared exhaustively. The parameters used are n the parameter of the gram structure, sw is the size of an n-gram sliding window to be considered. Parameter K represents the minimum number of common n-grams to increase a counter indicator. Finally, parameter C is the number of cores used in a parallelized implementation of the algorithm.

The second algorithm, "SimTF", is equivalent to algorithm 3.3, but the sorting strategy is based on *term frequency*. In this case it is expected a faster running time than the latter, at a cost of a possibly loss of recall because of the approximated nature of the approach. Then, "SimVP" is the algorithm 3.3 whose pseudo-code is presented in section 3. In this case, as well as "SimTF", it is expected a faster running time than "SimParalell" at a cost of a possibly loss of recall.

Finally, the "Diff" approach is a basic algorithm based on the `unix` command `diff`. This approach is based on the move, delete and add characteristics presented by the command, where each one of these outputs is used to determine the scoring function for plagiarism detection.

All of these algorithms outputs are considered as an approximation of the plagiarism detection problem, for which further analysis needs to be taken into consideration for a given pair of documents. They do not offer the offset nor the length of the plagiarism passages, however they determine how close a pair of documents are.

4.2 Evaluation Criteria

The resulting confusion matrix of this binary classification task can be described using four possible outcomes: Correctly classified plagiarized documents or True Positives (TP), correctly classified non plagiarized documents or True Negative (TN), wrong classified non plagiarized documents as plagiarized or False Positive (FP), and wrong classified plagiarized documents as non-plagiarized or False Negative (FN).

The evaluation criteria considered are common information retrieval measures, which are constructed using the before mentioned classification outcomes.

– Precision, that states the degree in which a pair of documents identified as a plagiarism case have indeed copy between them, and Recall, that states the percentage of plagiarized documents that the classifier manages to classify correctly. Can be interpreted as the classifier's effectiveness.

$$\text{Precision} = \frac{TP}{TP + FP}\ , \text{Recall} = \frac{TP}{TP + FN} \tag{1}$$

Table 1. Algorithms and their parameters used for the conducted experiment.

Name	Description	Parameters
SimParalell0	CD Sim paralell original	$(n = 3, sw = 5, K = 3, c = 16)$
SimParalell1	CD Sim paralell modified 1	$(n = 2, sw = 6, K = 3, c = 16)$
SimParalell2	CD Sim paralell modified 2	$n = 4, sw = 8, K = 3, c = 16$
SimTF0	CD Sim TF original	$(n = 3, sw = 5, \theta_1 = 7, \theta_2 = 2, k = 150)$
SimTF1	CD Sim TF modified 1	$(n = 2, sw = 6, \theta_1 = 7, \theta_2 = 2, k = 50)$
SimTF2	CD Sim TF modified 2	$(n = 4, sw = 8, \theta_1 = 7, \theta_2 = 2, k = 150)$
SimVP0	CD Sim AR original	$(n = 3, sw = 5, \theta_1 = 18, \theta_2 = 5, k = 250)$
SimVP1	CD Sim AR modified 1	$(n = 2, sw = 6, \theta_1 = 18\theta_2 = 5, k = 250)$
SimVP2	CD Sim AR modified 2	$(n = 4, sw = 8, \theta_1 = 18, \theta_2 = 5, k = 250)$
Diff0	CD Diff original	(Add = -1 , Move = 10 , Delete = -1)
Diff1	CD Diff modified 1	(Add = -10 , Move = 0 , Delete = -10)
Diff2	CD Diff modified 2	(Add = -5 , Move = 0 , Delete = -10)

– F-measure, the harmonic mean between the precision and recall, and Accuracy, the overall percentage of correct classified documents.

$$\text{F-measure} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}, \text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

5 Results and Discussions

Previous algorithms were evaluated using the evaluation criteria on the selected corpus from the PAN'09 dataset. All results are presented in table 2, where the accuracy, precision, recall, F-measure and the evaluation runtime are listed. The overall evaluation was performed for each plagiarized case, where for a given suspicious document, the confusion matrix was determined and their performance measures were evaluated. Then, after all suspicious documents were evaluated, the mean performance was recorded and listed in table 2.

Table 2. Results for Accuracy, Precision, Recall, F-measure and runtime for each algorithm presented in section 4

Copy Detector	Accuracy	Precision	Recall	F-measure	runtime (s)
SimParalell0	**0.999**	0.895	0.914	**0.904**	20,568
SimParalell1	0.990	0.616	0.958	0.750	21103
SimParalell2	0.961	**0.882**	0.916	0.899	29,655
SimTF0	0.874	0.824	0.821	0.823	6,959
SimTF1	0.923	0.766	0.800	0.783	7,451
SimTF2	0.874	0.836	0.818	0.827	6,615
SimVP0	0.887	0.865	0.857	0.861	5,393
SimVP1	0.899	0.859	0.852	0.855	5,596
SimVP2	0.849	0.828	0.868	0.847	**5,231**
Diff0	0.584	0.005	0.349	0.010	6,617
Diff1	0.007	0.007	**1.000**	0.014	6,529
Diff2	0.584	0.005	0.349	0.010	6,179

The results for the experiment are listed in Table 2. As the numbers indicate, the best results in term of F-measure are obtained with "SimParalell". This comes to no surprise, as the algorithm exhaustively checks the documents. The cost of such results, however, is the worst running time of the group. Alternatively, the "SimTF" and "SimVP" both get acceptable results but much better running times than "SimParalell". This factor is important as the number of pair of documents to compare becomes increasingly high. The "Diff" variant gets an overall worse result; based on the `diff` unix command entirely this approach does not take into account different obfuscation levels.

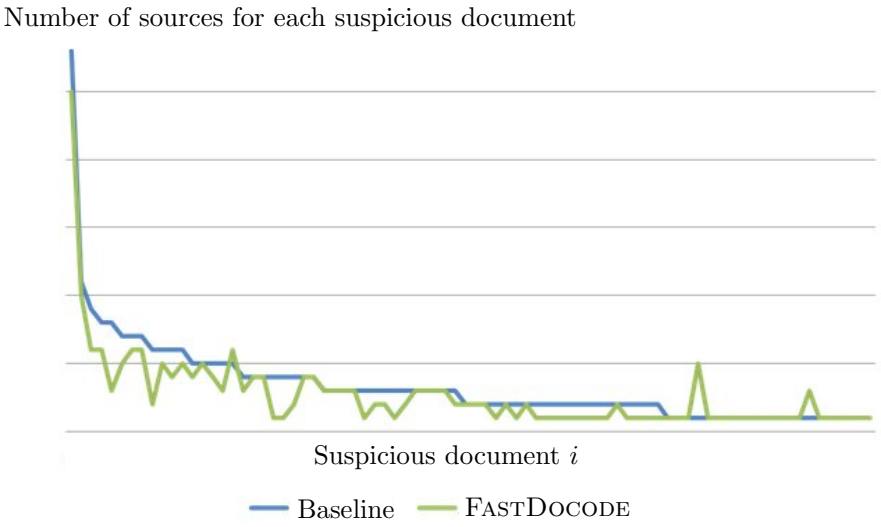

Figure 2. Results comparing the baseline sources for suspicious documents (blue line), and those retrieved by FastDocode (green line).

In Figure 2, results for the SimVP0 algorithm, where the expected curve for source-suspicious relationship is presented together with the source-suspicious relationship that was retrieved with the proposed algorithm. These results show that in the overall evaluation of the selected corpus, our proposal was robust in different number of sources for each suspicious evaluated.

6 Conclusions

In this work we have presented a method for uncovering external plagiarism cases. The strategy proposed is based on word tri-grams and word bi-grams, and consists basically on two phases. The first is aimed at reducing the search space for possible sources, and the second is aimed at exhaustively search a pair of document for plagiarized passages, where the offset and its length are computed.

While reducing the search space, we proposed a method that uses a statistical approach; removing stopwords and selecting samples based on alphabetic order, which helps to reduce considerably the running time of the algorithm. This proved to be empirically successful but further analysis must be taken into consideration.

Second, all algorithms parameters used were not selected using an extensive analysis on the algorithms performance; due to the size of the corpus it was difficult to run an optimization or grid search strategy over these parameters. We did, however, approximate them by iterating and trying on our sample, thus obtaining acceptable results.

As future work, it could be interesting to experiment the proposed approach with char n-grams instead of word n-grams. This could help FastDOCODE to include an intrinsic evaluation of a given document, or help the algorithm to detect plagiarized passages with high obfuscation levels.

7 Acknowledgment

Authors would like to thank continuous support of “Instituto Sistemas Complejos de Ingeniería” (ICM: P-05-004- F, CONICYT: FBO16; www.sistemasdeingenieria.cl); FONDEF project (DO8I-1015) entitled, DOCODE: Document Copy Detection (www.docode.cl); and the Web Intelligence Research Group (wi.dii.uchile.cl).

References

[1] Jun-Peng Bao, Jun-Yi Shen, Xiao-Dong Liu, Hai-Yan Liu, and Xiao-Di Zhang. Semantic sequence kin: A method of document copy detection. In Honghua Dai, Ramakrishnan Srikant, and Chengqi Zhang, editors, *PAKDD*, volume 3056 of *Lecture Notes in Computer Science*, pages 529–538. Springer Berlin / Heidelberg, 2004.

[2] Alberto Barrón-Cedeño. On the mono- and cross-language detection of text reuse and plagiarism. In *SIGIR '10: Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 914–914, New York, NY, USA, 2010. ACM.

[3] Alberto Barrón-Cedeño, Chiara Basile, Mirko Degli Esposti, and Paolo Rosso. Word length n-grams for text re-use detection. In *CICLing '10: Proceedings of the 11th International Conference on Computational Linguistics and Intelligent Text Processing*, pages 687–699, Berlin, Germany, 2010. Springer Berlin / Heidelberg.

[4] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *COLT '92: Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, New York, NY, USA, 1992. ACM.

[5] Felipe Bravo-Marquez, Gastón L'Huillier, Sebastián A. Ríos, Juan D. Velásquez, and Luis A. Guerrero. Docodelite: A meta-search engine for document similarity retrieval. In R. Setchi et al., editor, *KES'2010: 14th International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, pages 93–102., Berlin Heidelberg, 2010. Springer Berlin / Heidelberg.

[6] Sergey Brin, James Davis, and Héctor García-Molina. Copy detection mechanisms for digital documents. In *SIGMOD '95: Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, pages 398–409, New York, NY, USA, 1995. ACM.

[7] Zdenek Ceska. The future of copy detection techniques. In *Proceedings of the 1st Young Researchers Conference on Applied Sciences (YRCAS 2007)*, pages 5–107, Pilsen, Czech Republic, November 2007.

[8] Zdenek Ceska. Plagiarism detection based on singular value decomposition. In *GoTAL '08: Proceedings of the 6th international conference on Advances in Natural Language Processing*, pages 108–119, Berlin, Heidelberg, 2008. Springer Berlin / Heidelberg.

[9] Tommy W. S. Chow and M. K. M. Rahman. Multilayer som with tree-structured data for efficient document retrieval and plagiarism detection. *Trans. Neur. Netw.*, 20(9):1385–1402, 2009.

[10] C. Grozea, C. Gehl, and M. Popescu. Encoplot: Pairwise sequence matching in linear time applied to plagiarism detection. In Benno Stein, Paolo Rosso, Efstathios Stamatatos, Moshe Koppel, and Eneko Agirre, editors, *SEPLN 2009 Workshop on Uncovering Plagiarism, Authorship, and Social Software Misuse (PAN 09)*, pages 10–18. CEUR-WS.org, September 2009.

[11] Russell Hunt. Let's hear it for internet plagiarism. *Teaching Learning Bridges*, 2(3):2–5, 2003.

[12] K.O. Jones, J.M.V. Reid, and R Bartlett. Student plagiarism and cheating in an it age. In *Proceedings of the International Conference on Computer Systems and Technology*, pages IV.8:1–6, 2005.

[13] Bao Jun-Peng, Shen Jun-Yi, Liu Xiao-Dong, Liu Hai-Yan, and Zhang Xiao-Di. Document copy detection based on kernel method. In *Proceedings of the 2003 International Conference on Natural Language Processing and Knowledge Engineering.*, pages 250–255, 2003.

[14] NamOh Kang, Alexander Gelbukh, and SangYong Han. Ppchecker: Plagiarism pattern checker in document copy detection. In Petr Sojka, Ivan Kopecek, and Karel Pala, editors, *Text, Speech and Dialogue*, volume 4188 of *Lecture Notes in Computer Science*, pages 661–667. Springer Berlin / Heidelberg, 2006.

[15] T. Kohonen, M. R. Schroeder, and T. S. Huang, editors. *Self-Organizing Maps*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2001.

[16] Caroline Lyon, Ruth Barrett, and James Malcolm. A theoretical basis to the automated detection of copying between texts, and its practical implementation in the ferret plagiarism and collusion detector. In *Proceedings of Plagiarism: Prevention, Practice and Policies Conference*, Newcastle, UK, 2004.

[17] Caroline Lyon, James Malcolm, and Bob Dickerson. Detecting short passages of similar text in large document. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, pages 118–125, Pennsylvania, 2001.

[18] H. Maurer, F. Kappe, and B. Zaka. Plagiarism - a survey. *Journal of Universal Computer Science*, 12(8):1050–1084, 2006. | http://www.jucs.org/jucs_12_8/plagiarism_a_surveyl.

[19] Yurii Palkovskii. "counter plagiarism detection software" and "counter counter plagiarism detection" methods. In Benno Stein, Paolo Rosso, Efstathios Stamatatos, Moshe Koppel, and Eneko Agirre, editors, *SEPLN 2009 Workshop on Uncovering Plagiarism, Authorship, and Social Software Misuse (PAN 09)*, pages 67–68. CEUR-WS.org, September 2009.

[20] C Park. In other (people's) words: plagiarism by university students – literature and lessons. In *Assessment and Evaluation in Higher Education*, number 5, pages 471–488. Carfax Publishing, 2003.

[21] Martin Potthast, Alberto Barrón-Cedeño, Benno Stein, and Paolo Rosso. Cross-language plagiarism detection. *Language Resources and Evaluation*, pages 1–18, 2010.

[22] Martin Potthast, Benno Stein, Andreas Eiselt, Alberto Barrón-Cedeño, and Paolo Rosso. Overview of the 1st international competition on plagiarism detection. In Benno Stein, Paolo Rosso, Efstathios Stamatatos, Moshe Koppel, and Eneko Agirre, editors, *SEPLN 2009 Workshop on Uncovering Plagiarism, Authorship, and Social Software Misuse (PAN 09)*, pages 1–9. CEUR-WS.org, September 2009.

[23] Saul Schleimer, Daniel S. Wilkerson, and Alex Aiken. Winnowing: local algorithms for document fingerprinting. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 76–85, New York, NY, USA, 2003. ACM.

[24] Narayanan Shivakumar and Hector Garcia-Molina. Building a scalable and accurate copy detection mechanism. In *DL '96: Proceedings of the first ACM international conference on Digital libraries*, pages 160–168, New York, NY, USA, 1996. ACM.

[25] Antonio Si, Hong Va Leong, and Rynson W. H. Lau. Check: a document plagiarism detection system. In *SAC '97: Proceedings of the 1997 ACM symposium on Applied computing*, pages 70–77, New York, NY, USA, 1997. ACM.

[26] G. M. LaBeff E. E Vandehey, M. A. Diekhoff. College cheating: A twenty-year follow-up and the addition of an honor code. *Journal of College Student Development*, pages 468–480, 2007.

[27] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory (Information Science and Statistics)*. Springer Berlin / Heidelberg, 1999.

[28] Sven Meyer zu Eissen, Benno Stein, and Marion Kulig. Plagiarism detection without reference collections. In Reinhold Decker and Hans-Joachim Lenz, editors, *GfKl*, Studies in Classification, Data Analysis, and Knowledge Organization, pages 359–366. Springer Berlin / Heidelberg, 2006.