Plagiarism Detection with Genetic-Based Parameter Tuning

Article *in* International Journal of Pattern Recognition and Artificial Intelligence · June 2017

DOI: 10.1142/S0218001418600066

CITATIONS
2

READS
266

4 authors:

Miguel A. Sanchez-Perez
Instituto Politécnico Nacional
10 PUBLICATIONS **161** CITATIONS
SEE PROFILE

Alexander Gelbukh
Instituto Politécnico Nacional
529 PUBLICATIONS **5,837** CITATIONS
SEE PROFILE

Grigori Sidorov
Instituto Politécnico Nacional
226 PUBLICATIONS **1,745** CITATIONS
SEE PROFILE

Helena Gomez Adorno
Universidad Nacional Autónoma de México
47 PUBLICATIONS **531** CITATIONS
SEE PROFILE

Preprint of an article submitted for consideration in the International Journal of Pattern Recognition and Artificial Intelligence

Plagiarism Detection with Genetic-Based Parameter Tuning

Miguel A. Sanchez-Perez, Alexander Gelbukh, Grigori Sidorov and Helena Gómez-Adorno

Instituto Politécnico Nacional,
Centro de Investigación en Computación,
Av. Juan de Dios Bátiz, Esq. Miguel Othón de Mendizábal
Mexico City, Mexico
masp1988@hotmail.com, www.gelbukh.com, sidorov@cic.ipn.mx, helena.adorno@gmail.com

A crucial step in plagiarism detection is text alignment. This task consists in finding similar text fragments between two given documents. We introduce an optimization methodology based on genetic algorithms to improve the performance of a plagiarism detection model by optimizing its input parameters. The implementation of the genetic algorithm is based on non-binary representation of individuals, elitism selection, uniform crossover, and high mutation rate. The obtained parameters setting allow the plagiarism detection model to achieve better results than the state-of-the-art approaches.

Keywords: Plagiarism detection; text alignment; genetic algorithms; optimization.

1. Introduction

Plagiarism detection, and more generally, text reuse detection, has gained a lot of attention in the last few years. There is an increasing amount of digital information being produced on a daily basis. The easy access to the web, large databases, and telecommunication in general, has made it simpler to plagiarize or reuse the work of others. This issue has become a major problem for publishers, researchers, and educational institutions [20]. Plagiarism detection techniques are also useful in applications such as content authoring systems like Wikipedia, which offer fast and straightforward means for adding and editing content, and where avoiding content duplication is desired [3]. Other applications include author profiling [11, 17, 18] and author attribution [12]. Hence, detecting text reuse has become imperative in such contexts.

To promote studies on plagiarism detection related tasks, the PAN evaluation campaign[1] on uncovering plagiarism, authorship, and social software misuse, which is held as part of the CLEF conference, has been organized since 2009. It is constantly gaining much attention of researchers from around the world. The plagiarism detection task was divided into source retrieval and text alignment subtasks. In the text alignment subtask, the systems were required to identify all contiguous maximal-length passages of reused text between a given pair of documents.

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

In this paper, we present our approach to the text alignment subtask at PAN, which introduces a genetic algorithm to optimize the plagiarism detection system proposed in [31]. This optimization algorithm facilitates the discovery of optimal parameter settings, taking into consideration the combination of all the parameters that the approach needs as input. The use of a genetic algorithm to optimize the parameters allowed us to outperform the results of the best-performing systems of both PAN 2013 [27] and PAN 2014 [26] competitions.

The rest of the paper is organized as follows. In Section 2, we present related work in the field of text alignment and genetic algorithms applied to plagiarism detection. In Section 3, we briefly describe our general approach to text alignment. In Section 4, we define the genetic operators and the adaptation of our text alignment model as a fitness function. In Section 5, we describe the corpus used to optimize our approach and the hyper-parameters of the genetic algorithm. In Section 6, we analyze the obtained results and discuss our findings. Finally, in Section 7, we give conclusions and future work directions.

2. Related Work

The text alignment task, which is a subtask of the plagiarism detection task, consists in identifying similar passages of text in a given pair of documents, e.g., paragraphs copied verbatim or paraphrased.

There are several models of text alignment in the literature, the majority of them following a three-step approach: seeding, extension, and filtering [26].

The *seeding* step consists in finding relations (so-called *seeds*) between features extracted from the documents. Seeding approaches usually find relations either by exact match [9, 13, 30] or by creating matches replacing parts of text with other linguistics units [1]. The most common feature types are word n-grams with several implementations like context n-grams [23, 29, 34, 37], context skip n-gram [29], stopwords n-grams [34, 37] and named entity n-grams [34].

The *extension* step is the core of a text alignment model. It consists in joining the seeds into larger fragments, which become maximal length aligned text passages between the two documents. This step aims at identifying a plagiarized passage as a whole rather than only its fragments. Rule-based approaches have been the most popular strategy for extension algorithms [2, 8], although there are other kinds of methods based on dynamic programming [8, 22] and clustering [9, 13, 33].

The *filtering* step removes all aligned passages that do not meet certain criteria. Usually, this includes removing too short plagiarism cases [2, 9] or treating overlapping cases [33]. Although this assumption does not hold in real-world plagiarism detection, the majority of plagiarism cases in the PAN 2014 training corpus are contained inside larger cases in any of the two sides, either source or suspicious.

Table 1 summarizes the main ideas employed by the systems participating in PAN 2012 and 2013 [7, 14, 15, 23, 29, 34, 37].

In the majority of works on plagiarism detection, the parameters of the pro-

Table 1. Main ideas used in the systems participating in PAN 2012 and 2013. Table borrowed from [31].

Stage	**Method**	**[14]**	**[29]**	**[37]**	**[34]**	**[23]**	**[15]**	**Our**
Preprocessing	Special character removal	+	−	−	−	−	−	+
	Number removal	−	−	−	−	+	−	−
	Stopword removal	+	+	−	−	−	−	−
	Case folding	+	+	+	+	+	−	+
	Stemming	+	+	−	−	+	−	+
Seeding	Bag of words	+	−	−	−	−	+	+
	Context n-grams	−	+	+	+	+	−	−
	Context skip n-grams	−	+	−	−	−	−	−
	Stopword n-grams	−	−	+	+	−	−	−
	Named entity n-grams	−	−	−	+	−	−	−
Extension	Bilateral Alternating Sorting	+	−	−	−	−	−	−
	Distance between seeds	+	+	+	+	−	+	+
	Euclidean distance clusters	−	−	−	−	+	−	−
	Extension with multiple features	−	+	−	+	−	−	−
Filtering	Passage similarity	+	−	−	−	−	−	+
	Small passage removal	−	+	+	−	+	−	+
	Overlapping removal	−	−	+	+	−	−	+
	Nearby passage joining	−	−	−	+	−	−	−

posed models are determined through brute force [30,33] or by the author's domain knowledge [14]. Brute force approaches attempt to test all combinations of parameters at once, which is unfeasible if the approach has an extensive set of parameters. The majority of plagiarism detection algorithms apply brute force approaches for smaller subsets of parameters, disregarding plenty of parameter combinations. On the other hand, the parameter setting based on the author's knowledge does not take into account unknown information contained in the data.

We are aware of only few research works that use meta-heuristic based on genetic algorithms for plagiarism detection. Lange and Mancoridis [16] introduced a genetic algorithm for source code plagiarism detection. They defined 18 source code metrics to characterize a developer style and identified the optimal combination of these metrics using a genetic algorithm. The nearest neighbor classifier was used to determine if a particular developer wrote a piece of code. The cited research paper reports that the system is capable of identifying the true author of a source code with 55% of accuracy. However, the main contribution of the paper is the reduction of the search time of the optimal metrics set from weeks to hours by using the genetic algorithm instead of a greedy search.

Bouronara et al. [4] presented an approach for automatic plagiarism detection in the world of mail service based on a machine-learning tool and genetic algorithms.

Their first approach is based on character n-gram for the representation of the texts and tf-idf as weighting scheme to calculate the importance of a term in the corpus and a combination of two machine-learning methods: C4.5 and KNN algorithms. Then, they simulated a meta-heuristic method based on genetic algorithms with some variations of the hyper-parameters. The method based on genetic algorithms improved their initial results by 8.1% in the F-measure score.

Our previous research works [31, 33] introduced a plagiarism detection system that extracts sentences and compares them in a Vector Space Model (VSM) using the cosine similarity alike [14]. It also uses the Dice coefficient as in [15] given that this measure favors an equal vocabulary distribution employed in the passages to be compared. For the extension step, our algorithm clusters together nearby seeds. A plagiarism case is defined by the edges of a cluster and not as a set of seeds, this allows for small gaps in the range of seeds, which can be part of the plagiarism case even if the seeding process did not detect them. When filtering overlapped cases, we proposed a measure of quality to decide which one to keep and which one to discard.

In this work, we introduce a genetic algorithm to optimize our plagiarism detection system [31], tailoring to specific kinds of obfuscation. This optimization allows us to approximate the optimal parameter setting, taking into consideration all the parameters at once. We also seek to reduce time by finding the optimal parameter for our plagiarism detection method. However, unlike Lange and Mancoridis' work [16], our search space is bigger, in the range of 24 trillion (because we are not representing the metrics as binary genes).

3. Text Alignment Framework

Our approach to text alignment as part of the plagiarism detection task was introduced in [31–33]. In this section, we only explain the seeding and extension components, which are the core of our system and where the parameters being optimized are used.

3.1. *Seeding*

We measure the similarity between two sentences by representing them as tf-idf vectors in a Bag-of-Words (BOW) model, treating sentences as separate "documents." The idf measure calculated in this way is called *isf* measure (inverse sentence frequency) to emphasize that it is calculated over sentences as units and not documents:

$$tf\,(t,s) = f\,(t,s)\,, \tag{1}$$

$$isf\,(t,D) = \log \frac{|D|}{|\{s \in D : t \in s\}|}, \tag{2}$$

$$w\,(t,s) = tf\,(t,s) \times isf\,(t,D)\,, \tag{3}$$

where for term frequency $tf(t, s)$ we simply used the number of occurrences $f(t, s)$ of the term t in the sentence s; D is the set of all sentences in both given documents, and $w(t, s)$ is the final weight of a term t of the sentence s in our BOW representation.

After we defined the weighting scheme and transformed all sentences into vectors in both documents, we compared each sentence in the suspicious document to each sentence in the source document.

Now we construct the desired set S of seeds as

$$S = \{(i, j) \mid \cos(susp_i, src_j) > th\_cos \wedge \operatorname{dice}(susp_i, src_j) > th\_dice\}, \quad (4)$$

where the two sentences are represented as vectors, cos is the cosine similarity and dice is the Dice coefficient:

$$\cos(susp_i, src_j) = \frac{susp_i \cdot src_j}{|susp_i| \times |src_j|}, \quad (5)$$

$$\operatorname{dice}(susp_i, src_j) = \frac{2\,|\delta(susp_i) \cap \delta(src_j)|}{|\delta(susp_i)| + |\delta(src_j)|}, \quad (6)$$

where $\delta(x)$ is the set of non-zero coordinates of a vector x, $|*|$ is the Euclidean length of a vector or the cardinality of a set, respectively, and th_cos and th_dice are some thresholds determined experimentally.

3.2. *Extension*

Given the set of seeds S, defined as pairs (i, j) of similar sentences, the task of the extension stage is to form larger text fragments that are similar between two documents. For this, some left-hand sentences of S are joint into maximal contiguous fragments of the suspicious document, and some right-hand sentences into maximal contiguous fragments of the source document, so that those large fragments be still similar.

We divide the extension process into two steps: (1) clustering and (2) validation. In the clustering step, we create text fragments by grouping the seeds that are not separated by more than a *gap* number of sentences. In our implementation, we sort and cluster the set of seeds by i (left, or suspicious, document) such that $i_n - i_{n+1} \leq susp\_gap$; then, for each of the resulting clusters, we sort and cluster the obtained set by j using a src_gap threshold (right or source document), alternating these steps until no new clusters are formed. The clusters with fewer than $minsize$ seeds are discarded. Since we use the parameters $susp\_gap$ and src_gap to cluster seeds into larger text fragments, some sentences in these fragments may have no similarity to any of the sentences in the corresponding fragment. Therefore, to avoid adding to much noise in the clustering step we validate that the similarity between the text fragments of the remaining clusters exceeds some threshold. If the similarity is less than the given threshold, we apply the extension stage using $susp\_gap - 1$ and $src\_gap - 1$ for this particular cluster. We will reduce the gaps at most to a min_susp_gap and min_src_gap values, respectively. If the any of the

minimum values is reached and the validation condition is not met, then the cluster is discarded.

A text fragment is defined as the collection of all the sentences among the seeds of a particular cluster. Given a cluster integrated by seeds of the form (i, j), then the text fragment in the suspicious document F_{susp} is the collection of all the sentences from the smallest i to the largest i in the cluster. Similarly, the corresponding text fragment in the source document F_{src} is the collection of all the sentences from the smallest j to the largest j in the cluster.

We measured the similarity between text fragments F_{susp} and F_{src} computing the cosine between their vectors:

$$similarity\left(F_{susp}, F_{src}\right) = \cos\left(\sum_{v \in F_{susp}} v, \sum_{v \in F_{src}} v\right), \tag{7}$$

where the vector representation of the fragments is done adding together the vectors corresponding to all sentences of F_{susp} and F_{src} respectively.

For details of our method, see Algorithm 1. The variable *side* indicates by which side the pairs are clustered: $+1$ means clustering by sentences of the suspicious document (i) and -1, by sentences of the source document (j). In the algorithm, we generalize the thresholds as *maxgap* and *minsize* but in the implementation, there are separate thresholds for each document, and they are used depending in which side we are clustering. The output of the Extension stage is a set of pairs of similar text fragments $\{(F_{susp}, F_{src}), \dots\}$ taken from the resulting clusters.

4. Parameter Tuning

With the objective of optimizing the parameters of our plagiarism detection system in mind, we implemented a genetic algorithm using the basic operations with the settings that we think best fitted our needs. Table 2 describes the parameters we are optimizing. Our implementation of the genetic algorithm starts with a randomly generated population, then applies a fitness function to every individual and finally through the crossover and mutation genetic operators produces a new population. We used the elitist selection when constructing a new population allowing the two best individuals to carry over to the next generation, unaltered; and thus, ensuring that the solution quality of the genetic algorithm does not decrease from one generation to the next. The rest of the population is generated through crossover and mutation. We stop iterating when the maximum number of generations is reached.

4.1. *Search Space*

During the construction of our plagiarism detection system, we extensively experimented with several parameter settings. Taking into account the obtained results and our experience in the field, we chose certain ranges for each parameter that likely include the optimal values. We believe that our plagiarism detection model

Algorithm 1: Extension algorithm

```
const minsize, minsim
Function extension(seeds, maxgap)
    clusters ← clustering(seeds,maxgap,+1)
    clusters ← validation(clus,maxgap)
    return clusters
Function clustering(seeds, maxgap, side)
    clusters ← clusters of seeds such that in each cluster, side-hand
     sentences form in the document fragments with at most
     maxgap-sentence gaps
    discard all c ∈ clusters such that |c| < minsize
    if |clusters| ≤ 1 then
        return clusters
    else
        result ← ∅
        foreach c ∈ clusters do
            result ← result ∪ clustering(c,maxgap,−side)
    return result
Function validation(clusters, maxgap)
    result ← ∅
    foreach c ∈ clusters do
        if similarity(F_susp(c), F_src(c)) < th_val then
            if maxgap > min_maxgap then
                result ← result ∪ extension(c,maxgap − 1)
        else
            result ← result ∪ { c }
    return result
```

using these ranges generalizes to other datasets. In Table 3, we show these domains. They define the search space for the genetic algorithm, which consists of 24,761,352,699,900 possible combinations.

4.2. *Crossover*

There are many ways the crossover operator may be applied depending on the application and nature of the parameters being optimized [21]. The most widely used operators are:

- Single-point crossover: A single crossover position is chosen at random, and the parts of two parents after the crossover position are exchanged to form two offspring.
- Two-point crossover: Two positions are chosen at random, and the segments between them are exchanged.

Table 2. Plagiarism detection system parameters being optimized

	Parameter	Description
1	th_cos	Threshold for the cosine similarity during seeding
2	th_dice	Threshold for the Dice coefficient during seeding
3	th_val	Threshold for cosine similarity during validation
4	src_size	Min number of sentences in source fragment
5	src_gap	Max gap between sentences in source fragment
6	src_gap_summary	src_gap for the summary detection method
7	min_src_gap	Min value src_gap can take after several iterations
8	susp_size	Min number of sentences in suspicious fragment
9	susp_gap	Max gap between sentences in suspicious fragment
10	susp_gap_summary	susp_gap for the summary detection method
11	min_susp_gap	Min value susp_gap can take after several iterations

Table 3. Gene's domains

Parameter	Domain		
	Min	Max	Step
th_cos	0.20	0.50	0.01
th_dice	0.20	0.50	0.01
th_val	0.20	0.50	0.01
src_size	1	3	1
src_gap	0	30	1
src_gap_summary	0	30	1
min_src_gap	0	9	1
susp_size	1	3	1
susp_gap	0	30	1
susp_gap_summary	0	30	1
min_susp_gap	0	9	1

- Uniform crossover: The exchange happens at each gene position with a probability p.

Single-point and two-point crossovers cannot represent certain schemes, and genes' ordering is relevant for them. Given that our parameters (genes) lack a natural ordering, we used the uniform crossover in our experiments. The probability p of selecting a gene from a parent is randomly assigned every time the crossover operator is called. We use only two parents, which are selected by their fitness function value $f(x_i)$: given a population of n individuals $x_1, x_2, \ldots, x_n$ the probability

Algorithm 2: Crossover parents selection

$\{ w_i \} \leftarrow \{ f(x_i) \}$
$T \leftarrow \sum_{i=1}^{n} w_i$
$r \leftarrow rand(0, T)$
for $i \leftarrow 1, \ldots, n$ **do**
  if $r < w_i$ **then**
    return x_i
  $r \leftarrow r - w_i$
return x_n

to select an individual x_i for crossover is defined as

$$T = \sum_{i=1}^{n} f(x_i), \tag{8}$$

$$p(x_i) = \frac{f(x_i)}{T}. \tag{9}$$

An efficient algorithm to implement the selection of the parents for crossover is given in Algorithm 2. This implementation represents a different equation,

$$p(x_i) = (1 - p_{i-1}) \left(\frac{f(x_i)}{T - f(x_{i-1})} \right). \tag{10}$$

Theorem 1. *The probability* (9) *of selecting an individual* x_i *and its implementation* (10) *are equivalent:*

$$\frac{f(x_i)}{T} = (1 - p_{i-1}) \left(\frac{f(x_i)}{T - f(x_{i-1})} \right). \tag{11}$$

Proof. For $x_0 = 0$, assume $f(x_0) = 0$. Consider the base case $i = 1$; then

$$\frac{f(x_1)}{T} = (1 - 0) \left(\frac{f(x_1)}{T - 0} \right) = \frac{f(x_1)}{T};$$

thus, the conclusion holds for $i = 1$. By the inductive hypothesis, assume that the conclusion holds for all values of i up to some k, $k \geq 1$, and consider $i = k + 1$. Then

$$\begin{aligned} \frac{f(x_{k+1})}{T} &= (1 - p_k) \frac{f(x_{k+1})}{T - f(x_k)} \\ &= \left(1 - \frac{f(x_k)}{T} \right) \frac{f(x_{k+1})}{T - f(x_k)} \\ &= \frac{\cancel{T - f(x_k)}}{T} \frac{f(x_{k+1})}{\cancel{T - f(x_k)}} \\ &= \frac{f(x_{k+1})}{T}, \end{aligned}$$

where the second line holds by the inductive hypothesis. □

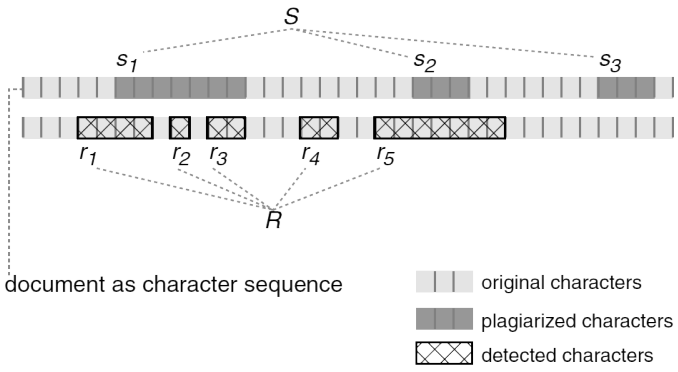

Figure 1. A document as character sequence, including plagiarized sections S and detections R returned by a plagiarism detection algorithm [28]

4.3. *Mutation*

We apply the mutation operator to each gene in every individual in the new population, except for those individuals that moved to the next generation through elitism selection. The mutation depends on a certain rate and when happening it changes the value of a parameter by randomly selecting one of the possible values for that particular parameter.

4.4. *Fitness Function*

The algorithm for calculating our fitness function consisted of two parts: running our plagiarism detection model explained in Section 3 and computing the *plagdet* metric, which returns a value between 0 and 1. The organizers of PAN introduced this metric in the context of their benchmarking workshop [25,28], which have been in use since 2009 and became the baseline in plagiarism detection evaluation.

4.4.1. *Evaluation Metric*

To keep the paper self-contained, we briefly explain the metrics. A document d is represented as a set of references to its characters $d = (1, d), ..., (|d|, d)$, where (i, d) refers to the $i - th$ character in d. A plagiarism case s is then represented as $s = s_{plg} \cup s_{src}$, where $s_{plg} \subseteq d_{plg}$ and $s_{src} \subseteq d_{src}$. Likewise, a detection r can be represented as $r = r_{plg} \cup r_{src}$. Figure 1 shows these representations which help us introduce the metrics. First, it is said that r detects s iff $r_{plg} \cap s_{plg} \neq \phi$ and $r_{src} \cap s_{src} \neq \phi$. Then, the precision and recall of R under S are defined as follows:

$$precision\,(S,R) = \frac{1}{|R|}\sum_{r\in R}\frac{\left|\bigcup_{s\in S}(s\sqcap r)\right|}{|r|} \tag{12}$$

$$recall\,(S,R) = \frac{1}{|S|}\sum_{s\in S}\frac{\left|\bigcup_{r\in R}(s\sqcap r)\right|}{|s|} \tag{13}$$

where

$$s\sqcap r = \begin{cases} s\cap r & \text{if } r \text{ detects } s,\\ 0 & \text{otherwise.}\end{cases} \tag{14}$$

Another concept that characterizes the power of a detection algorithm is whether a plagiarism case $s \in S$ is detected as a whole or in several pieces. Ideally, an algorithm should report detections R in a one-to-one manner to the true cases S. To capture this characteristic the detection granularity of R under S is defined as:

$$granularity\,(S,R) = \frac{1}{|S_R|}\sum_{s\in S_R}|R_s|\,, \tag{15}$$

where $S_R \subseteq S$ are cases detected by detections in R, and $R_s \subseteq R$ are the detections of a given s:

$$S_R = \{s|s\in S \wedge \exists r\in R : r \text{ detects } s\}\,, \tag{16}$$

$$R_s = \{r|r\in R \wedge r \text{ detects } s\}\,. \tag{17}$$

The domain of $granularity(S,R)$ is $[1,|R|]$, with 1 indicating the desired one-to-one correspondence and $|R|$ indicating the worst case, where a single $s \in S$ is detected over and over again.

Finally, precision, recall, and granularity are combined to an overall score *plagdet* (plagiarism detection):

$$plagdet\,(S,R) = \frac{F_\alpha}{\log_2\left(1+granularity\,(S,R)\right)}, \tag{18}$$

where F_α denotes the F_α-Measure, i.e., the weighted harmonic mean of precision and recall. At PAN workshops $\alpha = 1$ was used to compare the performance of the systems (precision and recall equally weighted) since there is currently no indication that either of the two is more important. The logarithm is applied to the granularity to decrease its impact on the overall score. The *plagdet* metrics represents the fitness value in our genetic algorithm.

4.4.2. *Improving the Running Time*

PAN 2014 organizers provided an on-line experimentation platform called TIRA [10] where participants were able to submit their systems. In the platform, authors were

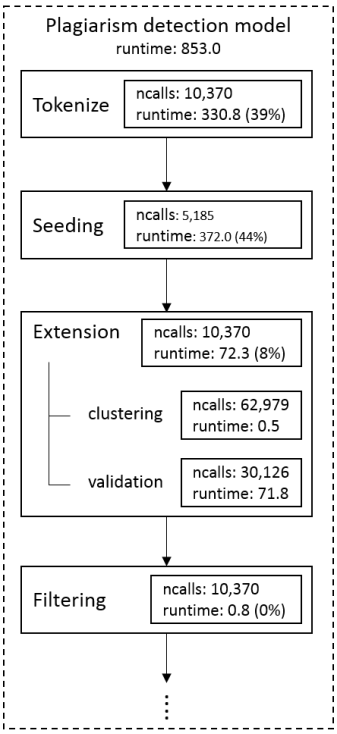

Figure 2. System components

provided a virtual machine with 1 processor and 4 GB of RAM memory. The running time of our system reported at [26] was of approx. 25 min, something that makes using the system as the fitness function in the genetic algorithm unfeasible. Therefore, we modified the system focusing on the main components of the implementation and its running time as shown in Figure 2, where the ellipsis stands for the rest of the system where running time was not relevant. The running time information was obtained using Python's profiler module.

First, we moved the *Tokenize* module, which ran the preprocessing of all the documents and loaded all document representations into memory, out of the genetic algorithm. Besides *Tokenize*, the most time-consuming module was *Seeding*, which had a complexity of $O(n^2)$, where n is the number of sentences in each document. To reduce this running time, we computed the seeds for each pair of documents only once using the lowest values of th_cos and th_dice outside the genetic algorithm and loaded the results into memory. Then, for each call to the fitness function, that called our plagiarism detection model, we filtered the seeds using the new th_cos and th_dice thresholds. We call this process *Seeds filtering*. Its complexity is still $O(n^2)$ for the worst case; however, the worst case is extremely unlikely because, given two documents d_1 and d_2, all sentences in d_1 should be similar to the rest of

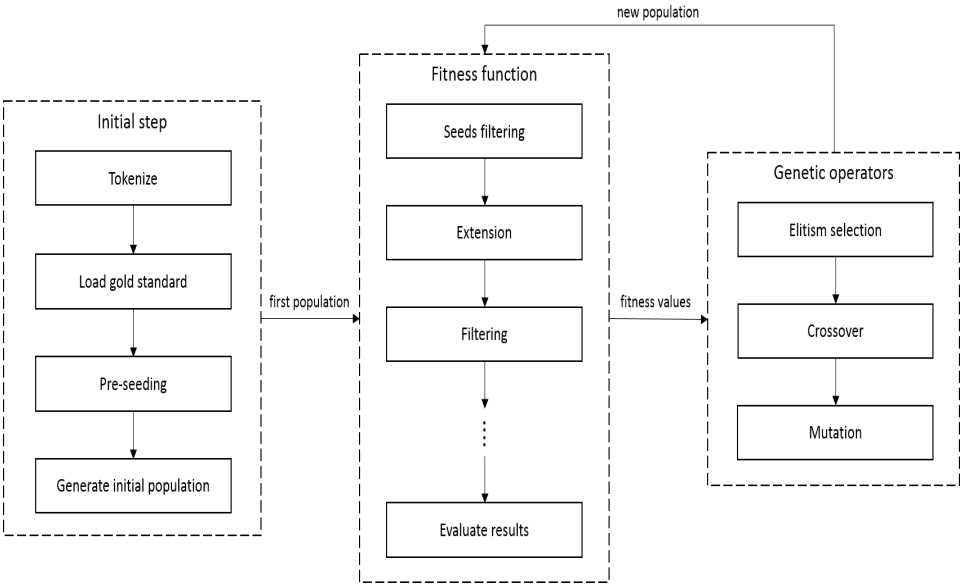

Figure 3. Implementation of the genetic algorithm

its sentences and all sentences in d_2, and vice-versa.

Figure 3 shows the diagram of the genetic algorithm. During our experiments, we called the fitness function 40,000 times when using the entire corpus and 20,000 times for each of the four sub-corpus.

5. Experimental Setting

Defined the genetic algorithm operators and the plagiarism detector optimizations, we established some hyper-parameters as shown in Table 4. The individual size is the number of parameters being optimized, and it is fixed. The population size and number of generations were chosen according to the amount of time we had to run tests using two computers with four processors each one. The uniform crossover ratio may have values between 0 and 1, raising the issue of not having the crossover at all for values close to 0 or 1, with a $\frac{2}{18} \approx 0.18$ probability of happening given the individual size used. In future work, we plan to use a different crossover ratio, although this issue did not affect negatively the results given the number of generations and the speed of convergence for this particular application and dataset. The mutation rate was chosen at 0.1, which is a rather large value. This rate was chosen because of the size of each individual, the non-binary representation, and the elitism selection; meaning that at least one gene is changed for every individual. We also have a huge searching space, and we keep the quality of the solution by passing the best individuals to the next generation unaltered.

For our experiments, we used the PAN 2014 training corpus, which is divided

Table 4. Genetic algorithm's hyper-parameters

Number of generations	1000
Population size	20
Individual size	11
Uniform crossover ratio p	$0 \leq p \leq 1$
Mutation rate	0.1

in five sub-corpus given the type of obfuscation used to generate each plagiarism case. We present a brief definition of each obfuscation type while a detailed description can be found at [27]. It is important to point out that plagiarism cases are corresponding text fragments that are small parts of a given pair of suspicious and source document, meaning we need to find then in a given context and it is not a classification task.

- **No plagiarism:** There are no plagiarism cases in this sub-corpus.
- **None:** Plagiarized fragments are a verbatim copy of the source.
- **Random:** Plagiarized fragments are a randomly obfuscated version of the source.
- **Translation:** Plagiarized fragments are generated by translating a source fragment through several languages and using different machine translators returning to the initial language.
- **Summary:** Plagiarized fragments are human-generated summaries of the sources.

Table 5 shows the corpus composition where *Pairs* represents the document pairs to be compared, *Pairs w/ PC* the pairs of documents with at least one plagiarism case and *PC* the plagiarism cases in the sub-corpus.

Table 5. PAN 2014 training corpus distribution

Sub-corpus	**Pairs**	**Pairs w/ PC**	**PC**
No-plagiarism	1000	0	0
None	1000	1000	1252
Random	1000	1000	1267
Translation	1000	1000	1250
Summary	1185	238	238
Entire corpus	5185	3238	4007

Intuitively, each type of obfuscation has different properties. Hence, proposing an algorithm capable of generalizing for each type of plagiarism with a fixed set of parameters is nearly impossible. In our previous approaches, we tried to address this issue. In PAN 2014 [31], we ran our model twice with different gap parameters, one

with *(src_gap, susp_gap)* and the other with larger parameters *(src_gap_summary, susp_gap_summary)*. The aim was to detect between the *summary* sub-corpus and the rest, where the ratio of the size of the plagiarized fragment and the corresponding source was a lot great in this sub-corpus. In [32], we incorporated a longest common substring method that ran over the results of our model and helped to improve the precision of our approach in the *none* sub-corpus dramatically, without hindering the performance of the others. The main idea of this work is to optimize the parameters of the basic model for each one of the sup-corpus, something that could be used later in conjunction with a rule-based [32] or machine learning classifier [19] to improve the performance of the plagiarism detection model.

6. Results and Discussion

Figure 4 shows the best individuals over all generations. We can see that all of the sub-corpus converges quickly to a stable optimal value around the 100th generation. This behavior reflects two things: first, our model generalizes well for the dataset at hand giving good results for a broad range of parameter values, mostly due to the robustness of the extension algorithm that adjusts some of the parameters dynamically. Second, the fact that the plagiarism cases are generated automatically for the majority of the cases and inserted randomly in a suspicious document increases the probabilities that the surrounding contexts to the plagiarism case are entirely unrelated, like a white box on a black background, allowing simple BOW models to easily detect the plagiarism cases.

The benefit of using a genetic algorithm to optimize the parameters of our model is that allows us to explore several regions of the immense search space taking into account all the parameters at once, something unfeasible to perform using a brute force approach. In Figure 5 and Figure 6 we plot the distribution of the parameters used as input to our plagiarism detection system as part of the fitness function, i.e. all the parameter values that were tested throughout the 1000 generations. The results show how the parameters converge to certain optimal values depending on the sub-corpus being used. This convergence happens due to the elitism and the way we select the parents for crossover, which keeps the parameter values close to a local optimum.

A deeper analysis of Figures 5 and 6 shows that for the *None* sub-corpus the threshold th_cos and th_dice, controlling the seeding component, converge to higher values while the th_val does not approach an specific value. This is something to expect given that the plagiarism cases in the *None* sub-corpus are verbatim copies of the sources and hence the similarity of their sentences is close to 1. Likewise, the susp_gap and src_gap tends to have the minimum value because consecutive sentences in plagiarized fragments have exact matches in the source. It is expected to get better results in this sub-corpus; however, the majority of the undetected cases or noise were due to sentence-splitting errors because of missing ending points, which in turn is caused by the random position where a plagiarized fragment was

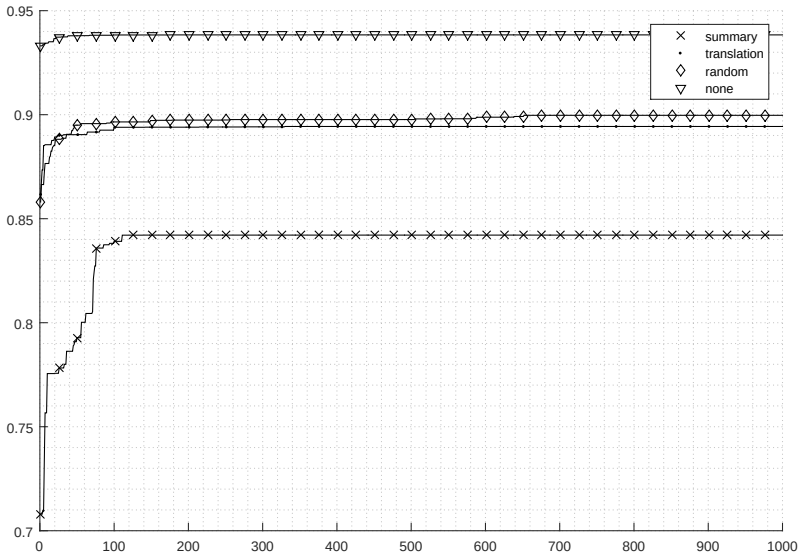

Figure 4. Best individual's fitness value by generation

inserted in a suspicious document when generating this sub-corpus.

The *Random* and *Translation* sub-corps results behave similarly given that both types of obfuscation simulate paraphrase by changing words with synonyms, sentence reordering, or cyclic translation. The similarity measures used in our model does not capture semantic equivalences found in paraphrased sentences, so it relies on lower similarity thresholds for the seeding stage and bigger validation threshold for the extension. An intuitive idea to improve the results for these plagiarism cases is using semantic similarity measures. However, an important note about these two sub-corpus is that in the case of randomly generated cases, the degree of obfuscation varies from a few random changes to a whole lot of them, generating text fragments without any sense to a human. Likewise, plagiarism cases generated through cyclic translation varies from using a pair of extensively translated languages to an abundance of unrelated pairs of languages and naive machine translation approaches, again producing meaningless text fragments.

Finally, in the *Summary* sub-corpus the results reflect the expected behavior of the parameters, where the similarity thresholds are close to the minimum values while the gaps between the source seeds are considerably larger than the suspicious counterpart. The low threshold values suggest that we should use summary detection methods for this kind of obfuscation instead of the traditional cosine similarity and Dice coefficient. The difference between the src_gap and susp_gap reflects the nature of a summary, which is smaller than the original text.

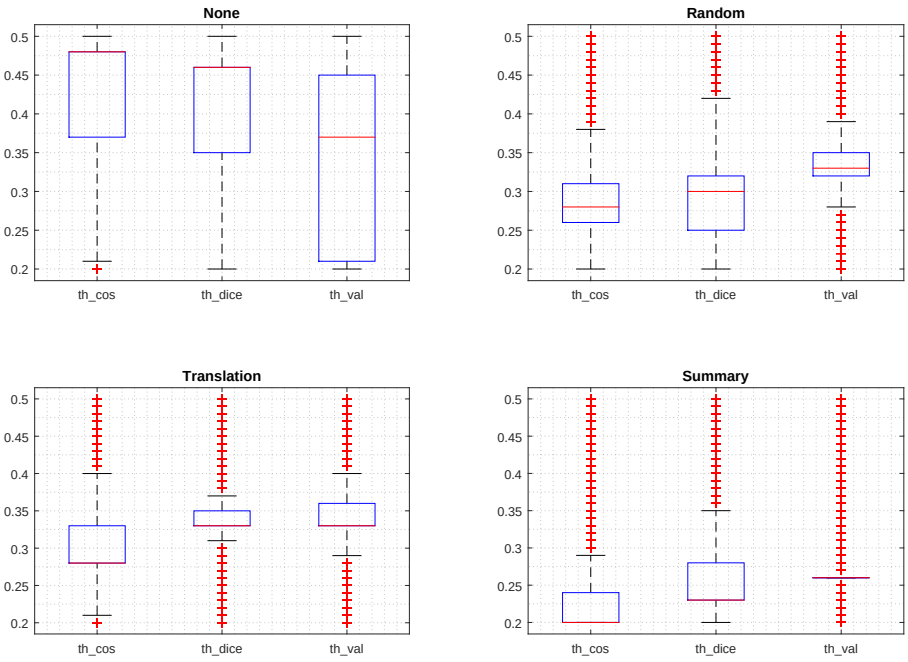

Figure 5. Similarity thresholds value's distribution

Besides running the genetic algorithm for each sub-corpus, we also optimized the parameters of our model with and without the summary heuristic and longest common substring method over the entire corpus (Genetic All & Genetic Simpler respectively). These experiments show some improvements over our previous parameter setting. Table 6 presents the parameters used at PAN and the final configurations resulting from the genetic algorithm for each one of the experiments.

Given the final parameters, we compare the results of each experiment in Table 7 by running our plagiarism detection system over the PAN 2014 test corpus. Values marked by an asterisk (*) were obtained by gathering the results of running our model over each sub-corpus individually knowing a priori the type of obfuscation and what parameter setting to use. To get these results, one would need to use a classifier capable of determining the type of obfuscation in a pair of documents and using the corresponding parameter setting.

When optimizing over the entire corpus (Genetic All & Genetic Simpler), we improved the results, even slightly, of our previous implementations (PAN All & PAN Simpler), which were already the best-performing methods in the PAN 2014 Text Alignment corpus [26].

As expected, specific obfuscation type optimization gave better results than those obtained with the fixed set of parameters, except for the *None* sub-corpus

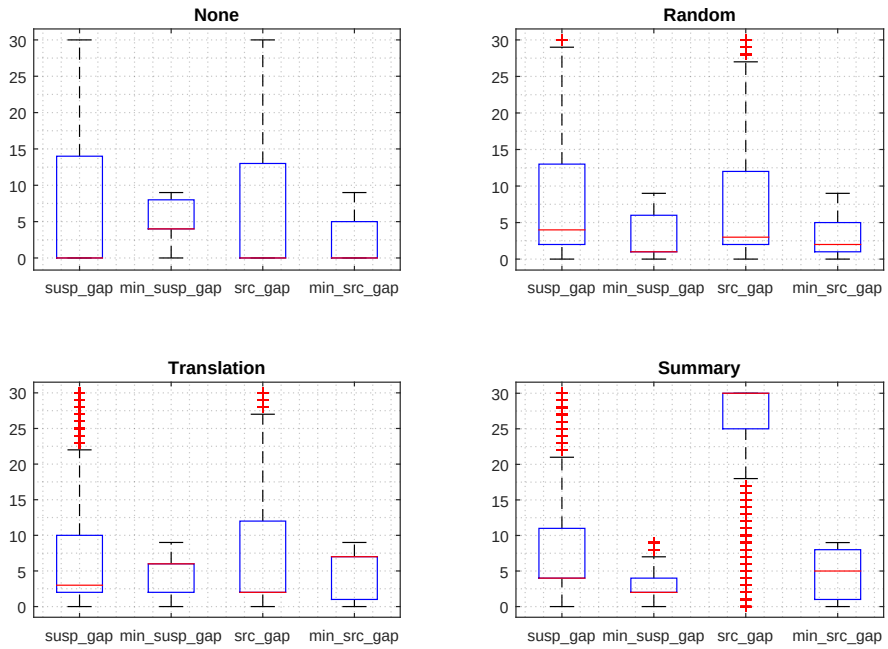

Figure 6. Extension parameters value's distribution

Table 6. Final parameters

Parameter	PAN	Genetic					
		Simpler	All	None	Random	Translation	Summary
th_cos	0.30	0.33	0.30	0.48	0.26	0.28	0.20
th_dice	0.33	0.39	0.31	0.46	0.25	0.33	0.23
th_val	0.34	0.22	0.34	0.45	0.33	0.33	0.26
src_size	1	1	1	1	1	1	1
src_gap	4	19	3	0	2	2	30
src_gap_summary	24	-	29	-	-	-	-
min_src_gap	0	9	3	0	1	2	8
susp_size	1	1	1	1	1	1	3
susp_gap	4	4	3	0	2	2	4
susp_gap_summary	24	-	28	-	-	-	-
min_susp_gap	0	0	3	0	1	2	2

which performed better in the approaches that use the Longest Common Substring (LCS) method. Hence, we also added the results of using the parameters of the genetic algorithm with the LCS method, which outperformed the rest of the exper-

Table 7. Plagiarism detection results using the final parameters over PAN 2014 test corpus. For the meaning of asterisk, see Section 6.

Approach	None	Random	Translation	Summary	Entire
PAN All	0.9854	0.8846	0.8751	0.6329	0.9010
Genetic All	**0.9859**	0.8841	0.8764	0.6493	0.9021
Genetic	0.9421	**0.8921**	**0.8940**	**0.8192**	0.9041*
Genetic + LCS	0.9732	0.8757	0.8936	0.8173	**0.9085***
PAN Simpler	**0.9010**	**0.8912**	**0.8868**	0.3108	0.8687
Genetic Simpler	0.8960	0.8751	0.8763	**0.6261**	**0.8733**

iments.

7. Conclusions and Future Work

We introduced a genetic algorithm (GA) to optimize the parameters of our plagiarism detection system improving our best-performing approaches presented at PAN shared tasks.

The genetic algorithm was implemented using a non-binary representation of individuals while adapting the genetic operators accordingly to work with defined parameter's domains. The elitism selection and weighted choice of crossover parents allowed us to inferred ranges of values for the parameters according to the type of obfuscation.

We made some adaptations to our model giving preference to the processing load over the memory consumption making it run 10x faster. These changes allowed us to run the fitness function 120,000 times in our experiments.

As future work, we planned to test the genetic algorithm using different hyper-parameters like uniform crossover probability, mutation rate, population size, selection method and stop condition.

Some improvements can be implemented to the plagiarism detection system regarding the extension module, such as decreasing the gap parameters by more than one by knowing the maximum gap in a cluster, thus aiming to reduce the running time, or decreasing src_gap and susp_gap independently, thus aiming to improve the performance of the model over plagiarism cases with text fragments with different sizes, e.g., for the summary obfuscation type. We are also considering using other feature representation such as syntactic n-grams [24, 36] and applying different similarity measures based on ontologies like soft-cosine combined with WordNet similarity [35] while testing over another corpus.

Acknowledgment

The work was partially supported by the Mexican Government: SNI, COFAA IPN, CONACYT project 240844, SIP-IPN projects 20171813, 20172008, and 20161947.

Bibliography

1. S. Abnar, M. Dehghani, H. Zamani and A. Shakery, Expanded n-grams for semantic text alignment, in *CLEF (Working Notes)* (2014) pp. 928–938.
2. F. Alvi, M. Stevenson and P. D. Clough, Hashing and merging heuristics for text reuse detection., in *CLEF (Working Notes)* (2014) pp. 939–946.
3. D. Bär, T. Zesch and I. Gurevych, Text reuse detection using a composition of text similarity measures, in M. Kay and C. Boitet (eds.), *COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, 8–15 December 2012, Mumbai, India* (Indian Institute of Technology Bombay, 2012) pp. 167–184.
4. H. A. Bouarara, A. Rahmani, R. M. Hamou and A. Amine, Machine learning tool and meta-heuristic based on genetic algorithms for plagiarism detection over mail service, in *Computer and Information Science (ICIS), 2014 IEEE/ACIS 13th International Conference on* (2014) pp. 157–162.
5. L. Cappellato, N. Ferro, M. Halvey and W. Kraaij (eds.), *Working Notes for CLEF 2014 Conference, Sheffield, UK, September 15–18, 2014 CEUR Workshop Proceedings* Vol. 1180, (CEUR-WS.org, 2014).
6. P. Forner, R. Navigli, D. Tufis and N. Ferro (eds.), *Working Notes for CLEF 2013 Conference, Valencia, Spain, September 23–26, 2013 CEUR Workshop Proceedings* Vol. 1179, (CEUR-WS.org, 2013).
7. L. Gillam, Guess again and see if they line up: Surrey's runs at plagiarism detection notebook for PAN at CLEF 2013, in Forner *et al.* [6].
8. L. Gillam and S. Notley, Evaluating robustness for "ipcress´´: Surrey's text alignment for plagiarism detection., in *CLEF (Working Notes)* (2014) pp. 951–957.
9. D. S. Glinos, A hybrid architecture for plagiarism detection., in *CLEF (Working Notes)* (2014) pp. 958–965.
10. T. Gollub, M. Potthast, A. Beyer, M. Busse, F. M. R. Pardo, P. Rosso, E. Stamatatos and B. Stein, Recent trends in digital text forensics and its evaluation - plagiarism detection, author identification, and author profiling, in P. Forner, H. Müller, R. Paredes, P. Rosso and B. Stein (eds.), *Information Access Evaluation. Multilinguality, Multimodality, and Visualization - 4th International Conference of the CLEF Initiative, CLEF 2013, Valencia, Spain, September 23–26, 2013. Proceedings*, *Lecture Notes in Computer Science* Vol. 8138 (Springer, 2013) pp. 282–302.
11. H. Gómez-Adorno, I. Markov, G. Sidorov, J.-P. Posadas-Durán, M. A. Sanchez-Perez and L. Chanona-Hernandez, Improving feature representation based on a neural network for author profiling in social media texts, *Computational Intelligence and Neuroscience* **2016** (2016) p. 2.
12. H. Gómez-Adorno, G. Sidorov, D. Pinto, D. Vilariño and A. Gelbukh, Automatic authorship detection using textual patterns extracted from integrated syntactic graphs, *Sensors* **16**(9) (2016) p. 1374.
13. P. Gross and P. Modaresi, Plagiarism alignment detection by merging context seeds., in *CLEF (Working Notes)* (2014) pp. 966–972.
14. L. Kong, H. Qi, C. Du, M. Wang and Z. Han, Approaches for source retrieval and text alignment of plagiarism detection notebook for PAN at CLEF 2013, in Forner *et al.* [6].
15. R. Küppers and S. Conrad, A set-based approach to plagiarism detection, in P. Forner, J. Karlgren and C. Womser-Hacker (eds.), *CLEF 2012 Evaluation Labs and Workshop, Online Working Notes, Rome, Italy, September 17–20, 2012*, *CEUR Workshop Proceedings* Vol. 1178 (CEUR-WS.org, 2012)
16. R. C. Lange and S. Mancoridis, Using code metric histograms and genetic algorithms

to perform author identification for software forensics, in *Proceedings of the 9th annual conference on Genetic and evolutionary computation* (2007) pp. 2082–2089.
17. I. Markov, H. Gómez-Adorno, J.-P. Posadas-Durán, G. Sidorov and A. Gelbukh, Author profiling with doc2vec neural networkbased document embeddings, in *Proceedings of the 15th Mexican International Conference on Artificial Intelligence (MICAI 2016). Lecture Notes in Artificial Intelligence* (Springer, In press)
18. I. Markov, H. Gómez-Adorno, G. Sidorov and A. Gelbukh, Adapting cross-genre author profiling to language and corpus, in *Proceedings of the CLEF* (2016) pp. 947–955.
19. F. Mashhadirajab and M. Shamsfard, A text alignment algorithm based on prediction of obfuscation types using SVM neural network, in P. Majumder, M. Mitra, P. Mehta, J. Sankhavara and K. Ghosh (eds.), *Working notes of FIRE 2016 - Forum for Information Retrieval Evaluation, Kolkata, India, December 7-10, 2016.*, *CEUR Workshop Proceedings* Vol. 1737 (CEUR-WS.org, 2016) pp. 167–171.
20. H. Maurer, F. Kappe and B. Zaka, Plagiarism – A survey, *Journal of Universal Computer Science* **12** (August 2006) 1050–1084.
21. M. Mitchell, *An introduction to genetic algorithms* (MIT Press, 1998).
22. G. Oberreuter and A. Eiselt, Submission to the 6th international competition on plagiarism detection, from innovand.io, chile (2014), in *CLEF (Working Notes)* (2014)
23. Y. Palkovskii and A. Belov, Using hybrid similarity methods for plagiarism detection notebook for PAN at CLEF 2013, in Forner *et al.* [6].
24. J. Posadas-Durán, H. Gómez-Adorno, I. Markov, G. Sidorov, I. Batyrshin, A. Gelbukh and O. Pichardo-Lagunas, Syntactic n-grams as features for the author profiling task, in *Working Notes of CLEF 2015—Conference and Labs of the Evaluation forum* (2015)
25. M. Potthast, A. Barrón-Cedeño, A. Eiselt, B. Stein and P. Rosso, Overview of the 2nd international competition on plagiarism detection, in M. Braschler, D. Harman and E. Pianta (eds.), *CLEF 2010 LABs and Workshops, Notebook Papers, 22–23 September 2010, Padua, Italy*, *CEUR Workshop Proceedings* Vol. 1176 (CEUR-WS.org, 2010)
26. M. Potthast, M. Hagen, A. Beyer, M. Busse, M. Tippmann, P. Rosso and B. Stein, Overview of the 6th International Competition on Plagiarism Detection, in Cappellato *et al.* [5], pp. 845–876.
27. M. Potthast, M. Hagen, T. Gollub, M. Tippmann, J. Kiesel, P. Rosso, E. Stamatatos and B. Stein, Overview of the 5th International Competition on Plagiarism Detection, in Forner *et al.* [6].
28. M. Potthast, B. Stein, A. Eiselt, A. Barrón-Cedeño and P. Rosso, Overview of the 1st international competition on plagiarism detection, in *In: SEPLN 2009 Workshop on Uncovering Plagiarism, Authorship, and Social Software Misuse (PAN 09), CEUR-WS.org* (2009) pp. 1–9.
29. D. A. Rodríguez Torrejón and J. M. Martín Ramos, Text alignment module in CoReMo 2.1 plagiarism detector notebook for PAN at CLEF 2013, in Forner *et al.* [6].
30. D. A. Rodríguez Torrejón and J. M. Martín Ramos, Coremo 2.3 plagiarism detector text alignment module, in *CLEF (Working Notes)* (2014) pp. 997–1003.
31. M. A. Sanchez-Perez, A. Gelbukh and G. Sidorov, Adaptive algorithm for plagiarism detection: The best-performing approach at PAN 2014 text alignment competition, in J. Mothe, J. Savoy, J. Kamps, K. Pinel-Sauvagnat, G. J. F. Jones, E. SanJuan, L. Cappellato and N. Ferro (eds.), *Experimental IR Meets Multilinguality, Multimodality, and Interaction - 6th International Conference of the CLEF Association, CLEF 2015, Toulouse, France, September 8–11, 2015, Proceedings*, *Lecture Notes in Computer Science* Vol. 9283 (Springer, 2015) pp. 402–413.
32. M. A. Sánchez-Pérez, A. Gelbukh and G. Sidorov, Dynamically adjustable approach

through obfuscation type recognition, in L. Cappellato, N. Ferro, G. J. F. Jones and E. SanJuan (eds.), *Working Notes of CLEF 2015—Conference and Labs of the Evaluation forum, Toulouse, France, September 8–11, 2015*, *CEUR Workshop Proceedings* Vol. 1391 (CEUR-WS.org, 2015)

33. M. A. Sanchez-Perez, G. Sidorov and A. Gelbukh, The winning approach to text alignment for text reuse detection at PAN 2014, in Cappellato *et al.* [5], pp. 1004–1011.
34. P. Shrestha and T. Solorio, Using a variety of n-grams for the detection of different kinds of plagiarism notebook for PAN at CLEF 2013, in Forner *et al.* [6].
35. G. Sidorov, A. Gelbukh, H. Gómez-Adorno and D. Pinto, Soft similarity and soft cosine measure: Similarity of features in vector space model, *Computación y Sistemas* **18**(3) (2014).
36. G. Sidorov, F. Velasquez, E. Stamatatos, A. Gelbukh and L. Chanona-Hernández, Syntactic dependency-based n-grams: More evidence of usefulness in classification, in A. Gelbukh (ed.), *Computational Linguistics and Intelligent Text Processing - 14th International Conference, CICLing 2013, Samos, Greece, March 24–30, 2013, Proceedings*, *Lecture Notes in Computer Science* Vol. 7816 (Springer, 2013) pp. 13–24.
37. S. Suchomel, J. Kasprzak and M. Brandejs, Diverse queries and feature type selection for plagiarism discovery notebook for PAN at CLEF 2013, in Forner *et al.* [6].