Detecting Plagiarism in Text Documents through Grammar-Analysis of Authors

Michael Tschuggnall, Günther Specht

Institute of Computer Science
Databases and Information Systems
Technikerstraße 21a
6020 Innsbruck
michael.tschuggnall@uibk.ac.at
guenther.specht@uibk.ac.at

Abstract: The task of intrinsic plagiarism detection is to find plagiarized sections within text documents without using a reference corpus. In this paper, the intrinsic detection approach Plag-Inn is presented which is based on the assumption that authors use a recognizable and distinguishable grammar to construct sentences. The main idea is to analyze the grammar of text documents and to find irregularities within the syntax of sentences, regardless of the usage of concrete words. If suspicious sentences are found by computing the pq-gram distance of grammar trees and by utilizing a Gaussian normal distribution, the algorithm tries to select and combine those sentences into potentially plagiarized sections. The parameters and thresholds needed by the algorithm are optimized by using genetic algorithms. Finally, the approach is evaluated against a large test corpus consisting of English documents, showing promising results.

1 Introduction

1.1 Plagiarism Detection

Today more and more text documents are made publicly available through large text collections or literary databases. As recent events show, the detection of plagiarism in such systems becomes considerably more important as it is very easy for a plagiarist to find an appropriate text fragment that can be copied, where on the other side it becomes increasingly harder to correctly identify plagiarized sections due to the huge amount of possible sources. In this paper we present the Plag-Inn algorithm, a novel approach to detect plagiarism in text documents that circumvents large data comparisons by performing intrinsic data anaylsis.

The two main approaches for identifying plagiarism in text documents are known as *external* and *intrinsic* algorithms [PEBC+11], where external algorithms compare a suspicious document against a given, unrestricted set of source documents like the world wide web, and intrinsic methods inspect the suspicious document only. Often applied techniques

used in external approaches include n-grams [OLRV10] or word-n-grams [Bal09] comparisons, standard IR techniques like common subsequences [Got10] or machine learning techniques [BSL+04]. On the other side, intrinsic approaches have to comprise the writing style of an author in some way and use other features like the frequency of words from predefined word-classes [OLRV11], complexity analysis [SM09] or n-grams [Sta09, KLD11] as well to find plagiarized sections.

Although the majority of external algorithms perform significantly better than intrinsic algorithms by using the advantage of a huge data set gained from the Internet, intrinsic methods are useful when such a data set is not available. For example, in scientific documents that use information mainly from books which are not digitally available, a proof of authenticity is nearly impossible for a computer system to make. Moreover, authors may modify the source text in such a way that even advanced, fault-tolerant text comparison algorithms like the longest common subsequence [BHR00] cannot detect similarities. In addition, intrinsic approaches can be used as a preceding technique to help reduce the set of source documents for CPU- and/or memory-intensive external procedures.

In this paper the intrinsic plagiarism detection approach *Plag-Inn* (standing for *Plag*iarism Detection *Inn*sbruck) is described, which tries to find plagiarized paragraphs by analyzing the grammar of authors. The main assumption is that authors have a certain *style* in terms of the grammar used, and that plagiarized sections can be found by detecting irregularities in their style. Therefore each sentence of a text document is parsed by its syntax, which results in a set of grammar trees. These trees are then compared against each other, and by using a Gaussian normal distribution function, sentences that differ significantly according to its building syntax are marked as suspicious.

The rest of this paper is organized as follows: the subsequent paragraph explains and summarizes the intrinsic plagiarism detection algorithm *Plag-Inn*, whereby Section 2 describes in detail how suspicious sentences are selected. The optimization of the parameters used in the algorithm is shown in Section 3 and an extensive evaluation of the approach is depicted in Section 4. Finally, Sections 5 and 6 discuss related work and conclude with the Plag-Inn algorithm by summarizing the results and showing future work, respectively.

1.2 The Plag-Inn Algorithm

The *Plag-Inn* algorithm is a novel approach (the main idea and a preliminary evaluation has been sketched in [TS12]) in the field of intrinsic plagiarism detection systems which tries to find differences in a document based on the stylistic changes of text segments. Based on the assumption that different authors use different grammar rules to build their sentences it compares the grammar of each sentence and tries to expose suspicious ones. For example, the sentence[1]

(1) *The strongest rain ever recorded in India shut down the financial hub of Mumbai, officials said today.*

[1] example taken and modified from the Stanford Parser website [Sta12]

could also be formulated as

(2) *Today, officials said that the strongest Indian rain which was ever recorded forced Mumbai's financial hub to shut down.*

which is semantically equivalent but differs significantly according to its syntax. The grammar trees produced by these two sentences are shown in Figure 1 and Figure 2, respectively. It can be seen that there is a significant difference in the building structure of each sentence. The main idea of the approach is to quantify those differences and to find outstanding sentences or paragraphs which are assumed to have a different author and thus may be plagiarized.

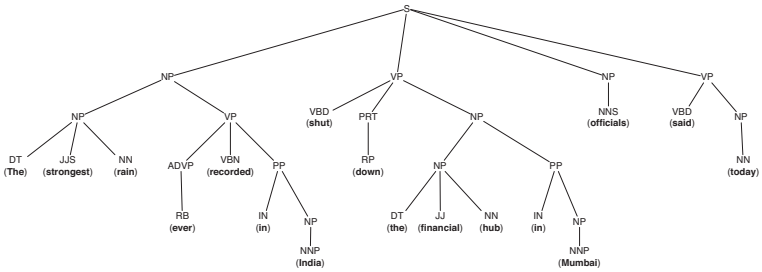


Figure 1: Grammar Tree Resulting From Sentence (1).

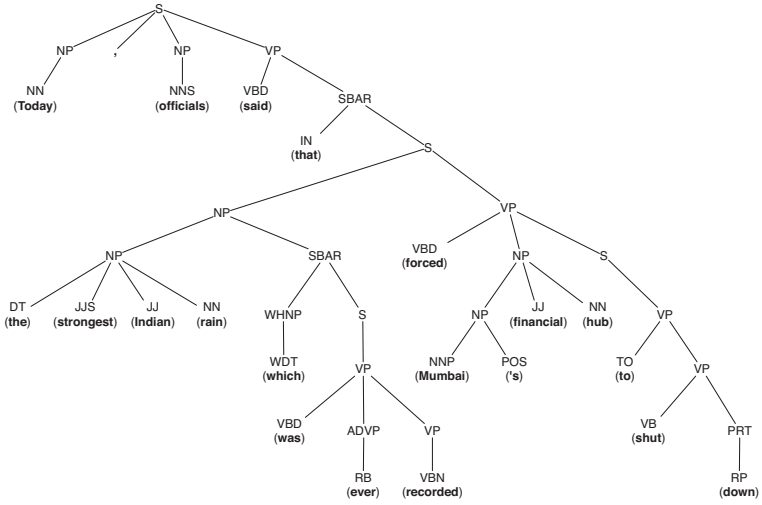


Figure 2: Grammar Tree Resulting From Sentence (2).

The Plag-Inn algorithm consists of five basic steps:

1. At first the given text document is parsed and split into single sentences by using Sentence Boundary Detection algorithms [SG00].

2. Then, the grammar is parsed for each sentence, i.e. the syntax of how the sentence was built is extracted. With the use of the open source tool *Stanford Parser* [KM03] each word is labelled with Penn Treebank tags [MMS93], that for example correspond to word-level classifiers like verbs (VB), nouns (NN) or adjectives (JJ), or phrase-level classifiers like noun phrases (NP) or adverbial phrases (ADVP). Finally, the parser generates a grammar syntax tree as it can be seen in e.g. Figure 1. Since the actual words in a sentence are irrelevant according to the grammatical structure, the leaves of each tree (i.e. the words) are dismissed.

3. Now, having all grammar trees of all sentences, the distance between each pair of trees is calculated and stored into a triangular distance matrix D:

$$D_n = \begin{pmatrix} d_{1,1} & d_{1,2} & d_{1,3} & \cdots & d_{1,n} \\ d_{1,2} & d_{2,2} & d_{2,3} & \cdots & d_{2,n} \\ d_{1,3} & d_{2,3} & d_{3,3} & \cdots & d_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ d_{1,n} & d_{2,n} & d_{3,n} & \cdots & d_{n,n} \end{pmatrix} = \begin{pmatrix} 0 & d_{1,2} & d_{1,3} & \cdots & d_{1,n} \\ * & 0 & d_{2,3} & \cdots & d_{2,n} \\ * & * & 0 & \cdots & d_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ * & * & * & \cdots & 0 \end{pmatrix}$$

   Thereby, each distance $d_{i,j}$ corresponds to the distance of the grammar trees between sentence i and j, where $d_{i,j} = d_{j,i}$. The distance itself is calculated using the pq-gram distance [ABG10], which is shown to be a lower bound of the more costly, fanout weighted tree edit distance [Bil05]. A pq-gram is defined through the base p and the stem q, which define the number of nodes taken into account vertically (p) and horizontally (q). Missing nodes, e.g. when there are less than q horizontal neighbors, are marked with $*$. For example, using $p = 2$ and $q = 3$, valid pq-grams of the grammar tree shown in Figure 1 would be $\{$*S-NP-NP-VP-**$\}$ or $\{$*S-VP-VBD-PRT-NP*$\}$ among many others. The pq-gram distance is finally calculated by comparing the sets PQ_i and PQ_j which correspond to the set of pq-grams of the grammar trees of the sentences i and j, respectively.

   The distance matrix of a document consisting of 1500 sentences is visualized in Figure 3, whereby the triangular character of the matrix is ignored in this case for better visibility. The z-axis represents the pq-gram-distance between the sentences on the x- and y-axis, and it can be seen that there are significant differences in the style of sentences around number 100 and 800, respectively.

4. Significant differences which are already visible to a human eye in the distance matrix plot are now examined through statistical methods. To find significantly outstanding sentences, i.e. sentences that might have been plagiarized, the median distance for each row in D is calculated. The resulting vector

$$\bar{d} = (\bar{d_1}, \bar{d_2}, \bar{d_3}, \ldots, \bar{d_n})$$

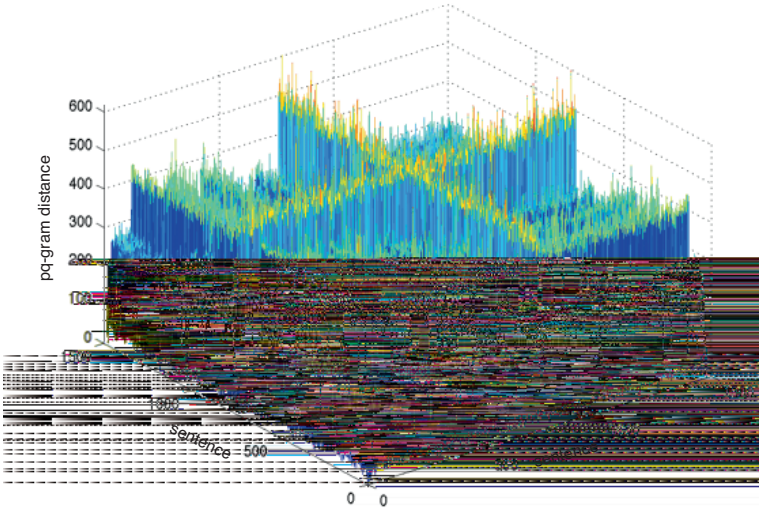


Figure 3: Distance Matrix of a Sample Document Consisting of about 1500 Sentences.

is then fitted to a Gaussian normal distribution which estimates the mean value μ and the standard deviation σ. The two Gaussian values can thereby be interpreted as a common variation of how the author of the document builds his sentences grammatically.

Finally, all sentences that have a higher difference than a predefined threshold δ_{susp} are marked as suspicious. The definement and optimization of δ_{susp} (where $\delta_{susp} \gg \mu + \sigma$) is shown in Section 3. Figure 4 depicts the mean distances resulting from averaging the distances for each sentence in the distance matrix D. After fitting the data to a Gaussian normal distribution, the resulting mean μ and standard deviation σ are marked in the plot. The threshold δ_{susp} that splits ordinary from suspicious sentences can also be seen, and all sentences exceeding this threshold are marked.

5. The last step of the algorithm is to smooth the results coming from the mean distances and the Gaussian fit algorithm. At first, suspicious sentences that are close together with respect to their occurence in the document are grouped into paragraphs. Secondly, standalone suspicious sentences might be dropped because it is unlikely in many cases that just one sentence has been plagiarized. Details on how sentences are selected for the final result are presented in Section 2.

2 Selecting Suspicious Sentences

The result of steps 1-3 of the Plag-Inn algorithm is a vector $\bar{d}$ of size n which holds the average distances of each sentence to all other sentences. After fitting this vector to a

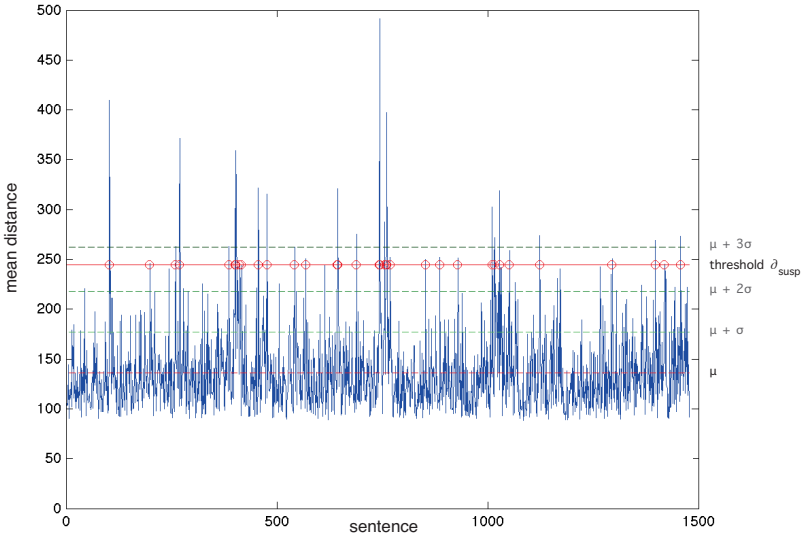


Figure 4: Mean Distances Including the Gaussian-Fit Values μ and σ.

Gaussian normal distribution as it is shown in Section 1.1, all sentences having a higher average distance than the predefined threshold δ_{susp} are marked as suspicious. This is the first step as can be seen in Algorithm 1.

The main objective of the further procedure of the sentence-selection algorithm is to group together sentences into plagiarized paragraphs and to eliminate standalone suspicious sentences. As the Plag-Inn algorithm is based on the grammatical structure of sentences, short instances like *"I like tennis."* or *"At what time?"* carry too less information and are most often not marked as suspicious as their building structure is too simple. Nevertheless, such sentences may be part of a plagiarized section and should therefore be detected. For example, if eight sentences in a row have found to be suspicious except one in the middle, it is intuitively very likely that it should be marked as suspicious as well.

To group together sentences, the algorithm shown in Algorithm 1 traverses all sentences in sequential order. If it finds a sentence that is marked suspicious, it first creates a new plagiarized section and adds this sentence. As long as ongoing suspicious sentences are found they are added to the section. When a sentence is not suspicious, the global idea is to use a lookahead variable ($curLookahead$) to step over non-suspicious sentences and check if there is a suspicious sentence *nearby*. If a sentence is then found to be suspicious within a predefined maximum ($maxLookahead$), this sentence and all non-suspicious sentences in between are added to the plagiarized section, and the lookahead variable is reset. Otherwise if this maximum is exceeded and no suspicious sentences are found, the current section is closed and added to the final result.

After all sentences are traversed, plagiarized sections in the final set R that contain only one sentence are checked to be filtered out. Intuitively this step makes sense as it can be assumed that authors do not copy only one sentence in a large paragraph. Within the evaluation of the algorithm described in Section 4 it could additionally be observed that single-sentence sections of plagiarism are often the result of wrongly parsed sentences coming from noisy data. To ensure that these sentences are filtered out, but *strongly* plagiarized single sentences remain in the result set, another threshold is introduced. In this sense, δ_{single} defines the average distance threshold that has to be exceeded by sections that contain only one sentence in order to remain in the result set R.

As the optimization of parameters (Section 3) showed, the best results can be achieved when choosing $\delta_{single} > \delta_{susp}$, which strengthens the intuitive assumption that a single-sentence section has to be *really* different. Controversially, genetic algorithms described in Section 3.2 also generated parameter optimization results that recommend to not filter single-sentence sections at all.

An example on how the algorithm works can be seen in Figure 5. Diagram (a) shows all sentences, where all instances with a higher mean distance than δ_{susp} have been marked as suspicious previously. When reaching suspicious sentence 5, it is added to a newly created section S. After adding sentence 6 to S, the lookahead variable is incremented as sentence 7 is not suspicious. Reaching sentence 8 which is suspicious again, both sentences are added to S as can be seen in Diagram (b). This procedure continues until the maximum lookahead is reached, which can be seen in Diagram (c). As the last sentence is not suspicious, S is closed and added to the result set R. Finally, Diagram (d) shows the final result set after eliminating single-sentence sections. As can be seen, sentence 14 has been filtered out as its mean difference is less than δ_{single}, whereas sentence 20 remains in the final result R.

3 Parameter Optimization

To evaluate and optimize the Plag-Inn algorithm including the sentence-selection algorithm, the PAN 2011 test corpus [PSBCR10] has been used which contains over 4000 English documents. The documents consist of a various number of sentences, starting from short texts from e.g. 50 sentences up to novel-length documents of about 7000 sentences. About 50% of the documents contain plagiarism, varying the amount of plagiarized sections per document, while the other 50% are left originally and contain no plagiarized paragraphs.

Most of the plagiarism cases are built by copying text fragments from other documents and subsequently inserting them in the suspicious document, while manual obfuscation of the inserted text is done additionally in some cases. Also, some plagiarism cases have been built using copying and translating from other source-languages like Spanish or German. Finally, for every document there exists a corresponding annotation file which can be consulted for an extensive evaluation.

As depicted in the previous section the sentence-selection algorithm relies on various input

Algorithm 1 Sentence Selection Algorithm

input:

d_i	mean distance of sentence i
δ_{susp}	suspicious sentence threshold
δ_{single}	single suspicious sentence threshold
$maxLookahead$	maximum lookahead for combining non-suspicious sentences
$filterSingles$	indicates whether sections containing only one sentence should be filtered out

variables:

$susp_i$	indicates whether sentence i is suspicious or not
R	final set of suspicious sections
S	set of sentences belonging to a suspicious section
T	temporary set of sentences
$curLookahead$	used lookaheads

set $susp_i \leftarrow false$ for all i, $R \leftarrow \emptyset, S \leftarrow \emptyset, T \leftarrow \emptyset$, $curLookahead \leftarrow 0$
for i from 1 to n **do**
    if $d_i > \delta_{susp}$ **then** $susp_i \leftarrow true$
end for

for i from 1 to number of sentences **do** ▷ traverse sentences
    if $susp_i = true$ **then**
        if $T \neq \emptyset$ **then**
            $S \leftarrow S \cup T$ ▷ add all non-suspicious sentences in between
            $T \leftarrow \emptyset$
        end if
        $S \leftarrow S \cup \{i\}$, $curLookahead \leftarrow 0$
    else if $S \neq \emptyset$ **then**
        $curLookahead \leftarrow curLookahead + 1$
        if $curLookahead \leq maxLookahead$ **then**
            $T \leftarrow T \cup \{i\}$ ▷ add non-suspicious sentence i to temporary set T
        else
            $R \leftarrow R \cup \{S\}$ ▷ finish section S and add it to the final set R
            $S \leftarrow \emptyset, T \leftarrow \emptyset$
        end if
    end if
end for
if $filterSingles = true$ **then** ▷ filter single-sentence sections
    for all plagiarized sections S of R **do**
        if $|S| = 1$ **then**
            $i \leftarrow$ the (only) element of S
            if $d_i < \delta_{single}$ **then** $R \leftarrow R \setminus \{S\}$
        end if
    end for
end if

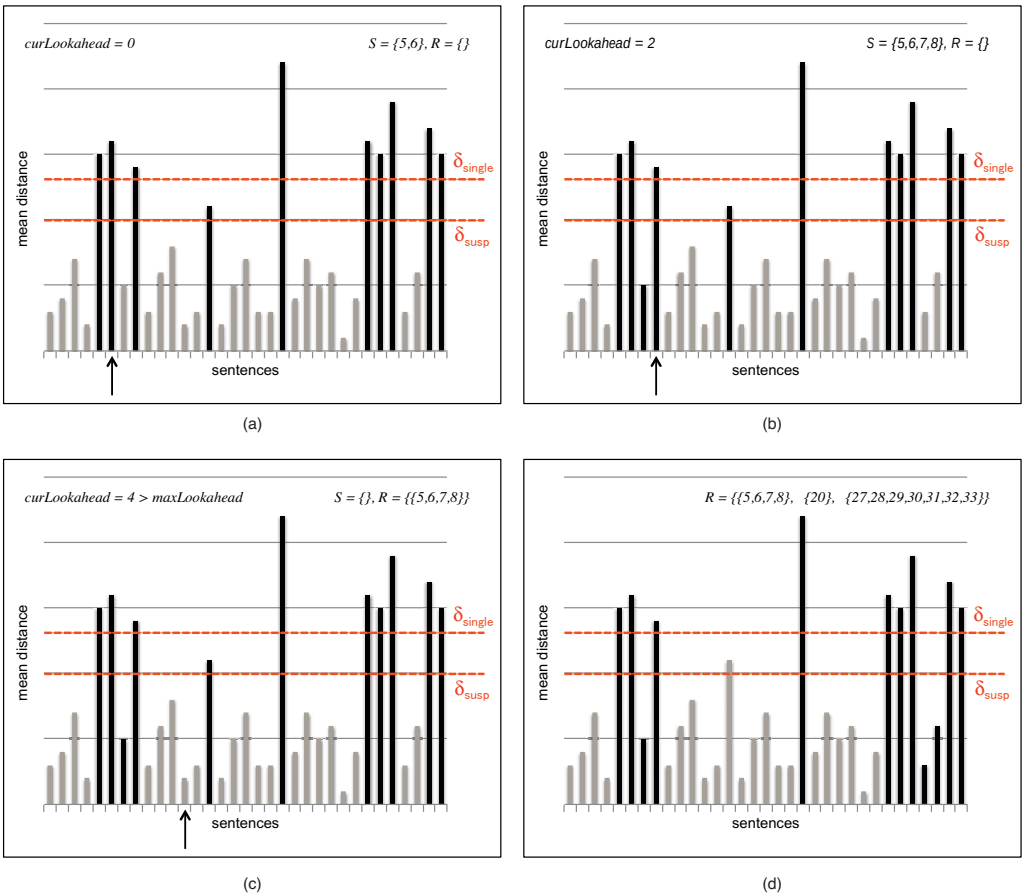


Figure 5: Example of the sentence-selection algorithm.

variables:

- δ'_{susp}: suspicious sentence threshold. Every sentence that has a higher mean distance is marked as suspicious.
- δ'_{single}: single suspicious sentence threshold. Every sentence in a single-sentence plagiarized section that is below this threshold is unmarked in the final step of the algorithm.
- $maxLookahead$: maximum lookahead. Defines the maximum value of checking if there is a suspicious sentence occuring after non-suspicious sentences that can be included into the current plagiarized section.
- $filterSingles$: boolean switch that indicates whether sections containing only one sentence should be filtered out. If $filterSingles = true$, the single suspicious

sentence threshold δ_{single} is used to determine whether a section should be dropped or not.

Thereby, the values for the thresholds δ'_{susp} and δ'_{single}, respectively, represent the inverse probability range of the Gaussian curve that include sentences with a mean distance that is not marked as suspicious. For example, $\delta'_{susp} = 0.9973$ would imply that $\delta_{susp} = \mu + 3\sigma$, meaning that all sentences having a higher average distance than $\mu + 3\sigma$ are marked as suspicious. In other words, one would find 99.73% of the values of the Gaussian normal distribution within a range of $\mu \pm 3\sigma$. In Figure 4, δ_{susp} resides between $\mu + 2\sigma$ and $\mu + 3\sigma$.

In the following the optimization techniques are described which should help finding the parameter configuration that produces the best result. To achieve this, predefined static configurations have been tested as well as genetic algorithms have been used. Additionally, the latter have been applied to find optimal configurations on two distinct document subsets that have been splitted by the number of sentences (see Section 3.3).

All configurations have been evaluated using the common IR-measures *recall*, *precision* and the resulting harmonic mean *F-measure*. In this case recall represents the percentage of plagiarized sections found, and precision the percentage of correct matches, respectively. In order to compare this approach to others, the algorithm defined by the PAN workshop [PSBCR10] has been used to calculate the according values[2].

3.1 Static Configurations

As a first attempt, 450 predefined static configurations have been created by building all[3] permutations of the values in Table 1.

Parameter	Range
δ'_{susp}	$[0.994, 0.995, \ldots, 0.999]$
δ'_{single}	$[0.9980, 0.9985, \ldots, 0.9995]$
$maxLookahead$	$[2, 3, \ldots, 16]$
$filterSingles$	$[\texttt{yes}, \texttt{no}]$

Table 1: Configuration Ranges for Static Parameter Optimization.

The five best evaluation results using the static configurations are shown in Table 2. It can be seen that all of the configurations make use of the single-sentence filtering, using almost the same threshold of $\delta'_{single} = 0.9995$. Surprisingly, the maximum lookahead with values

[2]Note that the PAN algorithms of calculating recall and precision, respectively, are based on plagiarized sections rather than plagiarized characters, meaning that if an algorithm detects 100% of a long section but fails to detect a second short section, the F-measure can never exceed 50%. Calculating the F-measure character-based throughout the Plag-Inn evaluation resulted in an increase of about 5% in all cases.

[3]In configurations where single-sentence sections are not filtered, i.e. $filterSingles = no$, permutations originating from the values of δ'_{single} have been ignored.

from 13 to 16 are quite high. Transformed to the problem definition and the sentence-selection algorithm, this means that the best results are achieved when sentences can be grouped together in a plagiarized section while stepping over up to 16 non-suspicious sentences.

As it is shown in the following section, genetic algorithms produced a better parameter configuration using a much lower maximum lookahead.

δ_{susp}	$maxLookahead$	$filterSingles$	δ_{single}	**Recall**	**Precision**	**F-Measure**
0.995	16	yes	0.9995	0.159	0.150	**0.155**
0.994	16	yes	0.9995	0.161	0.148	**0.155**
0.996	16	yes	0.9995	0.154	0.151	**0.153**
0.997	15	yes	0.9995	0.150	0.152	**0.152**
0.995	13	yes	0.9990	0.147	0,147	**0.147**

Table 2: Best Evaluation Results using Static Configuration Parameters.

3.2 Genetic Algorithms

Since the evaluation of the documents of the PAN corpus is computationally intensive, a better way than just evaluating fixed static configurations is to use genetic algorithms [Gol89] to find optimal parameter assignments.

Genetic algorithms emulate biological evolutions, implementing the principle of the *"Survival of the fittest"*[4]. In this sense genetic algorithms are based on *chromosomes* which consist of several *genes*. A gene can thereby be seen as a parameter, whereas a chromosome represents a set of genes, i.e. the parameter configuration. Basically, a genetic algorithm consists of the following steps:

1. Create a ground-population of chromosomes of size p.
2. Randomly assign values to the genes of each chromosome.
3. Evaluate the *fitness* of each chromosome, i.e. evaluate all documents of the corpus using the parameter configuration resulting from the individual genes of the chromosome.
4. Keep the fittest 50% of the chromosomes and alter their genes, i.e. alter the parameter assignments so that the population size is p again. Thereby the algorithms recognize whether a change in any direction lead to a fitter gene and takes it into account when altering the genes [Gol89].
5. If the predifined number of evolutions e is reached then stop the algorithm, otherwise repeat from step 3.

[4]The phrase was originally stated by the British philosopher Herbert Spencer.

With the use of genetic algorithms significantly more parameter configurations could be evaluated against the test corpus. Using the JGAP-library[5], which implements genetic programming algorithms, the parameters of the sentence-selection algorithm have been optimized. As the algorithm needs a high amount of computational effort and to avoid overfitting, random subsets of 1000 to 2000 documents have been used to evaluate each chromosome, whereby these subsets have been randomized and renewed for each evolution. As can be seen in Table 3 the results outperform the best static configuration with an F-measure of about 23%.

What can be seen in addition is that the best configuration gained from using a population size of $p = 400$ recommends to not filter out single-sentence plagiarized sections, but to rather keep them.

p	δ_{susp}	$maxLook.$	$filterSingles$	δ_{single}	**Recall**	**Precision**	**F**
400	0.999	4	no	-	0.211	0.257	**0.232**
200	0.999	13	yes	0.99998	0.213	0.209	**0.211**

Table 3: Parameter Optimization Using Genetic Algorithms.

3.3 Genetic Algorithms On Document Subsets

By a manual inspection of the individual results for each document of the test corpus it could be seen that in some configurations the algorithm produced very good results on short documents, while on the other hand it produced poor results on longer, novel-length documents. Additionally when using other configurations, the F-measure results of longer documents were significantly better.

To make use of the assumption that different length documents should be treated differently, the test corpus has been split by the number of sentences in a document. For example, when using 150 as splitting number, the subsets $S_{<150}$ and $S_{\geq 150}$ have been created, containing all documents that have less than 150 sentences and containing all documents that have more than or exactly 150 sentences, respectively. Then, for each of the two subsets, the optimal parameter configuration has been evaluated using genetic algorithms as it is described in Section 3.2. Like before, a random number of documents from 1000 to 2000 has been used to evaluate a chromosome.

Table 4 shows the best configurations produced by genetic algorithms using the sentence-split document subsets. For the dividing number 100, 150 and 200 sentences per document have been choosen as this seemed to be the optimal range found by manual inspection (as discussed in Section 6 this could be improved in future work).

With an F-measure of about 50% on the short-documents subset and 21% on the long-documents subset the sentence-split evaluation worked best with a splitting number of 100 and a resulting overall F-measure of 35.7%. All configurations significantly outperform

[5]http://jgap.sourceforge.net, visited October 2012

the genetic algorithm optimization described earlier, in the best case over 12%. Moreover, what can be seen in all configurations is that the short-documents subsets could be optimized significantly better, resulting in F-values of about 45% to 50%. As discussed later, this already indicates that the algorithm is well suited for short documents.

subset	δ_{susp}	$maxLook.$	$filterSingles$	δ_{single}	**Recall**	**Precision**	**F**
$S_{\geq 100}$	0.998	6	yes	0.9887	0.205	0.216	**0.210**
$S_{<100}$	0.999	1	no	-	0.501	0.508	**0.504**
					0.353	0.362	**0.357**
$S_{\geq 150}$	0.963	9	yes	0.9993	0.118	0.109	**0.113**
$S_{<150}$	0.999	4	no	-	0.494	0.478	**0.486**
					0.306	0.294	**0.300**
$S_{\geq 200}$	0.963	10	yes	0.9998	0.108	0.115	**0.111**
$S_{<200}$	0.999	2	yes	0.9999	0.441	0.457	**0.449**
					0.275	0.286	**0.280**

Table 4: Parameter Optimization Using Genetic Algorithms on Document Subsets Splitted by the Number of Sentences.

4 Evaluation

The Plag-Inn algorithm including the sentence-selection algorithm have been evaluated using the PAN 2011 test corpus which has been described in more detail in Section 3. It consists of more than 4000 English documents that randomly contain plagiarism cases.

The following section shows an extensive evaluation of the Plag-Inn approach using the sentence selection algorithm. All results are based on the optimal configuration gained from using the genetic algorithms over the whole test corpus as described in Section 3.2. Although the genetic algorithms optimized for the document subsets split by text length produced significant better results, it is more realistic to use just one parameter configuration in order to avoid overfitted results (manual inspection showed that small documents contained significantly less plagiarism cases in the test corpus).

Figure 6 shows the overall evaluation result with an F-measure of 23%[6]. It can be seen that plagiarism cases created by translation could be detected better than those created artificially[7]. This result is as expected because the grammar of another language is always different than the target language (English), and translation - which is done by automatic algorithms in most cases - produces changes in grammar that can be detected more easily.

The fourth column shows the results for documents where at least 75% of the plagiarism

[6] The currently (2012) best intrinsic plagiarism detector achieves an F-measure of about 33% over the same test corpus [OLRV11]. As shown, using the best parameter configuration on document subsets we achieve even 35%.

[7] *artificially* means that a source fragment has been copied and modified by computer algorithms.

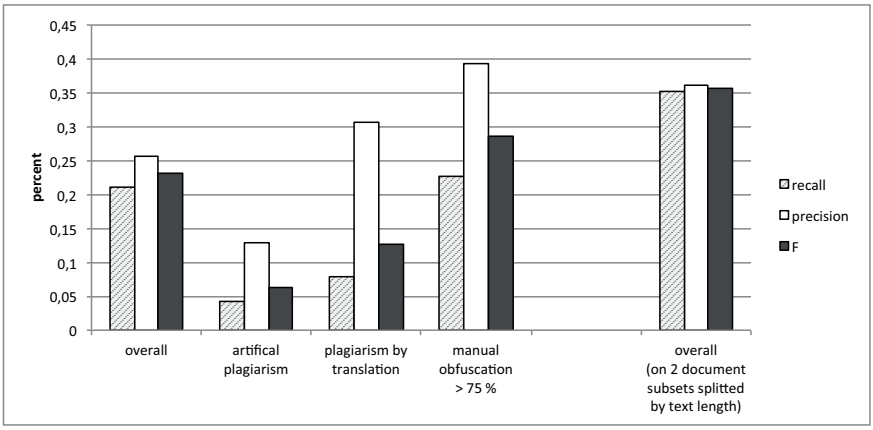


Figure 6: Overall Evaluation Results.

cases have been built by copying and subsequently doing manual obfuscation by hand. The F-measure for those cases almost reaches 30%, which indicates that the approach works very well for *real* plagiarism cases, i.e. cases that were created by humans rather than by computer algorithms. Finally, for comparison the results gained from the best parameter configurations on two document subsets splitted by the number of sentences contained is shown in the last column. As stated before, the best result of about 35% could be achieved by using a splitting number of 100.

In all results the precision is higher than the recall, meaning that the approach is better in interpreting suspicious passages than in finding them.

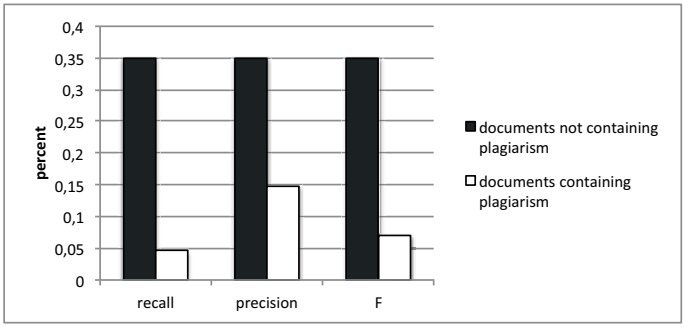


Figure 7: Evaluation Results for Documents Containing and Not Containing Plagiarism Cases.

The Plag-Inn approach achieves significantly different results on documents that contain plagiarism and on documents that do not contain plagiarism, as it is shown in Figure 7. Thereby, *clean* documents are processed very well and with balanced values for recall, precision and the according F-value of about 35%, respectively. On the other side, documents

containing plagiarized sections are obviously more difficult to detect for the algorithm, while the precision is consistently higher than the recall as experienced before.

Evaluations show that, the shorter documents get the better the algorithm works. Like it is illustrated in Figure 8, an F-measure of over 40% could be achieved on documents containing less than 300 sentences. Moreover as stated before, the algorithm performs steadily better when documents get shorter, reaching an F-measure of nearly 70% on very short documents containing less than 50 sentences. As the algorithm uses grammatical inconsistencies to find plagiarized sections, it might be that the variety of sentence syntaxes is too high in long documents, such that the algorithm fails frequently and produces the overall result of only 23%.

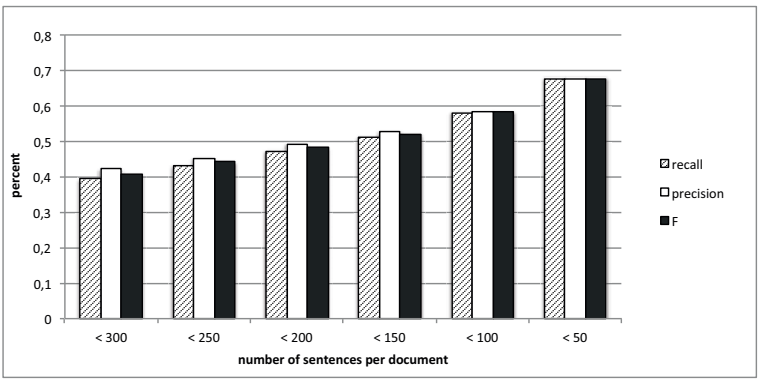


Figure 8: Evaluation Results for Short Documents.

What could be found in addition is that the approach is sensitive to the number of plagiarized sections per document as it is shown in Figure 9. Here, all documents that contain plagiarism have been inspected concerning the concrete number of plagiarism cases per document. It can be seen that the more sections of a document have been plagiarized, the better the results get. I.e. the more an author steals in his work, the more likely it is that it is detected by the algorithm.

Finally, the best option to improve the approach in future work is to reduce the number of false-positives, which is depicted in Figure 10. Diagram (a) shows the number of false-positives and false-negatives, respectively, where over 35% of all test corpus documents have wrongly been marked as plagiarized. On the horizontal axis the number of detected plagiarized sections for false-positives, and the number of not detected sections for false-negatives are shown. For about half of the wrongly predicted false-positives, the algorithm detected less than 5 plagiarism cases, and for about 10% of the documents the algorithm detected only one plagiarized section. This means that improving the algorithm in a way such that the false-positives that contain only one suspicious passage could be reduced or eliminated, this would lead to a significant better overall result.

In diagram (b) the percentage of predictions is shown. According to the high number of false-positives, the algorithm predicted too much for about half of the documents, i.e. it

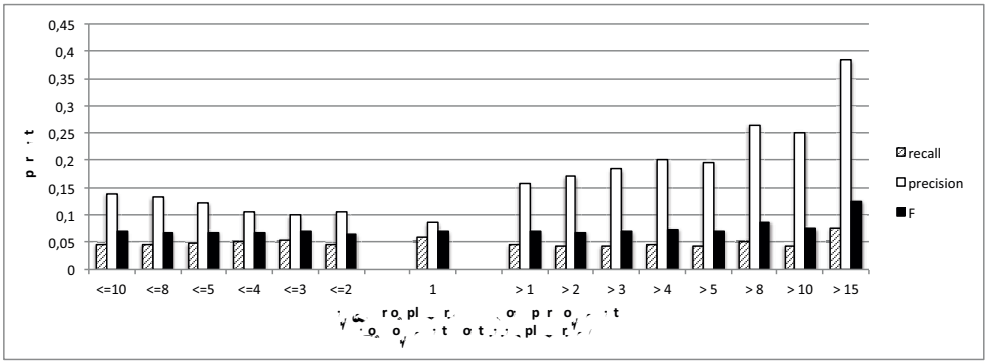
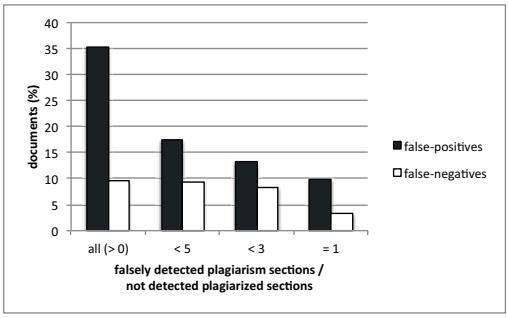
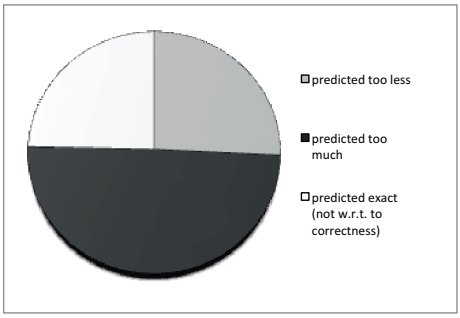


Figure 9: Evaluation Results Correlated to Number of Plagiarized Sections per Document.

detected plagiarized sections where there originally were less or even none. For the other half, too less or the exact number of sections have been detected. It has to be stressed that the amount of exact predictions in terms of plagiarized sections does not necessarily correspond to the number of *correct* detections. For example, the algorithm might have found four plagiarized sections in a document that contained exactly four plagiarized sections, but it might be the case that only two of them are correct. Nevertheless, manual inspections of the results have shown that the majority of exact predictions really correspond to the correct sections.



(a)



(b)

Figure 10: False-Positives, False-Negatives and General Prediction Statistics.

5 Related Work

An often used technique in intrinsic plagiarism detection algorithms are n-grams [Sta09, KLD11], where a text document is broken into sets of two-, three- or four-letter chunks

and subsequently analyzed by their number of occurences within sliding windows. With the additional use of a style change function [Sta09] could reach a recall and precision value of about 33% over the PAN11 test corpus. Another approach also uses the sliding window technique but is based on word frequencies and the assumption that authors use a significant set of words [OLRV11].

An approach that tries to recognize paraphrased sections based on the phrase structure of sentences and the structure of verb phrases is described in [UKN05]. In this work, sentence-initial and -final phrases are inspected together with predefined semantic classes of verbs [Lev93] by part-of-speech tagging. It uses POS-tags only, i.e. without referring to computationally intense full grammar tree parses.

[SM09] discusses the comparison of binary strings calculated from word groups like nouns or verbs using complexity analysis. Approaches in the field of author detection and genre categorization also use NLP tools to analyze documents based on syntactic annotations [SKF00], but do not use grammar trees for comparisons. Word- and text-based statistics like the average sentence length or the average parse tree depth are used in [Kar00].

6 Conclusion and Future Work

In this paper the intrinisic plagiarism detection approach *Plag-Inn* is described: it aims to find plagiarism in text documents by inspecting the suspicious document only. The main idea is to analyze grammatical structures that authors use to build sentences, and to find inconsistencies in the syntax. After inconsistencies have been found by using Gaussian normal distribution fitting, an algorithm that selects and combines suspicious sentences is presented.

Furthermore, various parameters of the algorithm have been optimized by using static configurations and genetic algorithms. Using the best parameter setting, the algorithm achieves an F-measure of about 23%. By additionally using different settings for short and long documents, an overall F-measure of about 35% could be achieved, which is a rather high value for intrinsic plagiarism detection systems. In addition, an F-score of over 50% could be gained for short documents. Thereby the splitting number for the two document subsets has intuitively been chosen, and it should be considered to find the optimal value algorithmically in future work.

Extensive evaluations showed that the approach works very well on short documents containing less than 300 sentences, and that the more authors plagiarize, the more likely it is for the algorithm to detect the according sections. For documents consisting of less than 50 sentences, an F-measure of nearly 70% could be reached. Nevertheless, a drawback of the approach is that it predicts too much in many cases - i.e. it detects plagiarism where there is none - leading to a high number of false-positives. Future work should concentrate on reducing this number to improve the algorithm significantly.

On the other side, the number of false-negatives is low, implying that the approach is well-suited for ensuring that a document is not plagiarized. Evaluations showed that if the algorithms states that a document is plagiarism-free, it is right in over 90% of the cases.

Other future work includes the application of the algorithm to other languages like German, French or Spanish, which could produce the same or even better results, because other languages often use more complex grammar syntaxes than English. The more choice an author has to build his sentences, the more individual *style* the documents gets, and the easier it should be for the Plag-Inn algorithm to find irregularities.

The approach could also be adapted to be able to verify and/or attribute authors of single-author text documents, or to verify authorships in multi-author text documents. Moreover, as the approach relies on the style of authors, it could be used for genre detection, spam filtering or recommender systems as well.

References

[ABG10] Nikolaus Augsten, Michael Böhlen, and Johann Gamper. The pq-Gram Distance between Ordered Labeled Trees. *ACM TRANSACTIONS ON DATABASE SYSTEMS (TODS)*, 2010.

[Bal09] Enrique Vallés Balaguer. Putting Ourselves in SME's Shoes: Automatic Detection of Plagiarism by the WCopyFind tool. In *Proceedings of the SEPLN'09 Workshop on Uncovering Plagiarism, Authorship and Social Software Misuse*, pages 34–35, 2009.

[BHR00] L. Bergroth, H. Hakonen, and T. Raita. A Survey of Longest Common Subsequence Algorithms. In *Proceedings of the Seventh International Symposium on String Processing Information Retrieval (SPIRE'00)*, SPIRE '00, pages 39–48, Washington, DC, USA, 2000. IEEE Computer Society.

[Bil05] Philip Bille. A survey on tree edit distance and related problems. *Theoretical Computuer Science*, 337:217–239, June 2005.

[BSL+04] 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, *Advances in Knowledge Discovery and Data Mining*, volume 3056, pages 529–538. Springer Berlin, Heidelberg, 2004.

[Gol89] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989.

[Got10] Thomas Gottron. External Plagiarism Detection Based on Standard IR Technology and Fast Recognition of Common Subsequences - Lab Report for PAN at CLEF 2010. In Martin Braschler, Donna Harman, and Emanuele Pianta, editors, *CLEF (Notebook Papers/LABs/Workshops)*, 2010.

[Kar00] Jussi Karlgren. *Stylistic Experiments For Information Retrieval*. PhD thesis, Swedish Institute for Computer Science, 2000.

[KLD11] Mike Kestemont, Kim Luyckx, and Walter Daelemans. Intrinsic Plagiarism Detection Using Character Trigram Distance Scores. In V. Petras, P. Forner, and P. Clough, editors, *CLEF 2011 Labs and Workshop, Notebook Papers*, Amsterdam, The Netherlands, 2011.

[KM03] Dan Klein and Christopher D. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 423–430, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[Lev93] Beth Levin. *English Verb Classes and Alternations : A Preliminary Investigation*. University Of Chicago Press, September 1993.

[MMS93] Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19:313–330, June 1993.

[OLRV10] Gabriel Oberreuter, Gaston L'Huillier, Sebastián A. Ríos, and Juan D. Velásquez. FastDocode: Finding Approximated Segments of N-Grams for Document Copy Detection. In Martin Braschler, Donna Harman, and Emanuele Pianta, editors, *Notebook Papers of CLEF 10 Labs and Workshops*, 2010.

[OLRV11] Gabriel Oberreuter, Gaston L'Huillier, Sebastián A. Ríos, and Juan D. Velásquez. Approaches for Intrinsic and External Plagiarism Detection - Notebook for PAN at CLEF 2011. In Vivien Petras, Pamela Forner, and Paul D. Clough, editors, *Notebook Papers of CLEF 11 Labs and Workshops*, 2011.

[PEBC+11] Martin Potthast, Andreas Eiselt, Alberto Barrón-Cedeño, Benno Stein, and Paolo Rosso. Overview of the 3rd International Competition on Plagiarism Detection. In Vivien Petras, Pamela Forner, and Paul Clough, editors, *Notebook Papers of CLEF 11 Labs and Workshops*, 2011.

[PSBCR10] Martin Potthast, Benno Stein, Alberto Barrón-Cedeño, and Paolo Rosso. An Evaluation Framework for Plagiarism Detection. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)*, Beijing, China, August 2010. Association for Computational Linguistics.

[SG00] Mark Stevenson and Robert Gaizauskas. Experiments on sentence boundary detection. In *Proceedings of the sixth conference on Applied natural language processing*, ANLC '00, pages 84–89, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

[SKF00] Efstathios Stamatatos, George Kokkinakis, and Nikos Fakotakis. Automatic text categorization in terms of genre and author. *Comput. Linguist.*, 26:471–495, December 2000.

[SM09] Leanne Seaward and Stan Matwin. Intrinsic Plagiarism Detection using Complexity Analysis. In *CLEF (Notebook Papers/Labs/Workshop)*, 2009.

[Sta09] Efstathios Stamatatos. Intrinsic Plagiarism Detection Using Character n-gram Profiles. In *CLEF (Notebook Papers/Labs/Workshop)*, 2009.

[Sta12] Stanford Parser Website. A statistical parser. http://nlp.stanford.edu/software/lex-parser.shtml, visited January 2012.

[TS12] Michael Tschuggnall and Günther Specht. Plag-Inn: Intrinsic Plagiarism Detection Using Grammar Trees. In Gosse Bouma, Ashwin Ittoo, Elisabeth Métais, and Hans Wortmann, editors, *NLDB*, volume 7337 of *Lecture Notes in Computer Science*, pages 284–289. Springer, 2012.

[UKN05] Özlem Uzuner, Boris Katz, and Thade Nahnsen. Using Syntactic Information to Identify Plagiarism. In *Proc. 2nd Workshop on Building Educational Applications using NLP. Ann Arbor*, 2005.