Expanded N-Grams for Semantic Text Alignment

Notebook for PAN at CLEF 2014

Samira Abnar, Mostafa Dehghani, Hamed Zamani, and Azadeh Shakery

School of ECE, College of Engineering,
University of Tehran, Tehran, Iran
{s.abnar, mo.dehghani, h.zamani, shakery }@ut.ac.ir
http://ece.ut.ac.ir/iis

Abstract Text alignment is a sub-task in the plagiarism detection process. In this paper we discuss our approach to address this problem. Our approach is based on mapping text alignment to the problem of subsequence matching just as previous works. We have prepared a framework, which lets us combine different feature types and different strategies for merging the features. We have proposed two different solutions to relax the comparison of two documents, so as to consider the semantic relations between them. Our first approach is based on defining a new feature type that contains semantic information about its corresponding document. In our second approach we have proposed a new method for comparing the features considering their semantic relations. Finally, We have applied DB-SCAN clustering algorithm to merge features in a neighborhood in both source and suspicious documents. Our experiments indicate that different feature sets are suitable for detecting different types of plagiarism.

Keywords: Plagiarism Detection, Text Alignment, Semantic Similarity, Expanded N-Gram, N-Gram Similarity Score, N-Gram Transparency, Dispersion Measure

1 Introduction

Text alignment is introduced as a phase of external plagiarism detection process [4,5]. It has many other applications such as parallel corpus construction [1]. In this paper, we discuss our approach for dealing with the text alignment challenge of PAN 2014 [3]. More generally, we present a mechanism for text alignment emphasizing on the application of plagiarism detection.

The aim of a text alignment tool is to detect pairs of related regions from two distinct documents. How these two regions are related and also the length of the regions should be determined regarding the application. Considering the plagiarism detection as an application of text alignment, on which we have focused in this paper, the relatedness between the regions would be not only in terms of concept and semantic, but also in terms of lexical and grammatical structures. Also about the length of the regions in case of the plagiarism detection, it is reasonable to be at least as long as a short paragraph; while in parallel corpus construction it may be fine for the length of the regions to be as long as a sentence.

At the first glance, the text alignment challenge seems similar to the problem of finding common subsequences of two sequences; thus each document is treated as a sequence of features. In the simplest way, we can reduce the problem to common substring matching. However, this will only work for finding no obfuscated plagiarism cases. Most of the methods that have been proposed until now, are based on this approach. Their difference is in the type and number of features exploited, and also the strategies they use so as to detect the subsequences, which indicate the plagiarism cases.

In our work, we have prepared a framework that enables us to customize the text aligner combining different common subsequence detection methods with different features sets. In our experiments we have tested a clustering based method as the subsequence detection strategy, and word n-grams, expanded word n-grams, contextual word n-grams [7] and stopword n-grams [6] as the features.

One of the goals we follow is to investigate how each of the phases of modeling the document as a sequence of features and finding the common subseqeces, affects the performance of the text aligner. As indicated by our experiments different combinations of values for n in n-grams are suitable for detecting different types of plagiarism.

The existing methods extract different kinds of features from the source and suspicious documents and try to detect plagiarism cases considering the common features. They assume a feature to be common, if it exactly appears in both documents. Thus, for example if a word n-gram is replaced completely by its synonym n-gram in the plagiarized text, it would not be considered as a common feature. However, in order to be able to detect more complicated types of plagiarism like summarization, we need to deal with some semantic feature types. Thus if we want to solve the problem in the existing frameworks there is the need to make use of many different other types of features. For example we may have to extend the document by adding the synonyms of its words to it. In this paper, we explore a new approach in finding common features to address this problem. We introduce a general similarity function, which assign a similarity score to feature pairs. This score can be based on many different aspects of the features rather than just comparing their string value. For instance, it may consider different forms of the n-gram tokens in the comparison. Then feature pairs with a similarity value higher than a specified threshold are considered as common features. Furthermore the assigned similarity score can be used later in the process of text alignment to give higher priority to the pairs with higher weights. The advantage of using a similarity function for comparing features is that there is no more needed to use many various types of features for detecting different types of plagiarism. Therefore, applying this approach, the word n-gram feature, itself, covers many other types of features. In section 2.2 we will discuss the similarity function we have exploited more thoroughly.

In Section 2 we will explain our methodology, then we will discuss the results we obtained in the Section 3. Finally we will conclude and discuss about some future works in Section 4.

2 Methodology

In this Section the overall procedure of the proposed method for text alignment is described thoroughly. The text alignment challenge we have dealt with, is defined as follows:

Definition 1 *Given two documents d and d', our goal is to extract a set of passage pairs, P, such that:*

$$P = \{< p_{di}, p_{d'j} > |\forall p_{di}, \forall p_{d'j} : p_{di} \in d \wedge p_{d'j} \in d' \wedge |p_{di} \cap p_{d'j}| > \delta\}, \quad (1)$$

where p_{di} is a passage from d, $p_{d'j}$ is a passage from d', and $p_{di} \cap p_{d'j}$ shows how two text regions are related. It can be defined in various ways. Finally, δ is a threshold that determines if the amount of relatedness of two text regions is as much as they should be considered as a plagiarism case.

Briefly, the proposed process to solve the text alignment challenge starts with extracting features from the documents. Then the features from two documents are compared so as to select the common features in order to align related sections of the texts. In the next step, for the purpose of detecting a subset of the common features, which are in a neighborhood in both documents, the feature pairs are considered as points in a two dimensional space according to their positions in the first and second documents. Then a density based clustering algorithm is applied to cluster these points. Thus each cluster represents a pair of related text regions in the first and second documents. By considering offsets of features in each cluster, boundaries of the related regions are determined.

In the rest of this section we explain all these steps in more details. In Section 2.1 we discuss how we have modeled each document as a set of features. Then in Section 2.2 we describe our approaches for detecting common features. Finally in Sections 2.3 and 2.4 we explain the techniques we have employed to approximately detect the plagiarism cases and then find the exact edges of the detected plagiarism cases respectively.

2.1 Extracting Features from Textual Documents

We have proposed two different methods for considering semantics in text alignment. In both these methods so as to be able to compare two documents for the text alignment purpose, each document is represented as a set of feature objects, $d = \{f_i\}$. Feature objects are defined as:

$$F =< t, s, S, P, T >, \quad (2)$$

where t is the feature type, s is the string value of the feature, S is a list containing the string value of the tokens included in the feature, P is a list of occurrence positions of the n-gram in the document, and T is a list that keeps the *n-grams transparency* in each position.

In the first method we use an exact technique of comparison on the string value of the features to detect common features. Therefore, it is needed to exploit multiple feature types to be able to detect the plagiarism cases with different levels of obfuscation. In this research, we have explored using word n-grams, contextual word n-grams, stopword n-grams, and *expanded word n-grams*. We have introduced the expanded word n-grams to be able to detect semantic relations between two documents. Algorithm 1 shows how expanded word n-grams are created.

```
**Data**: $g$ : $n$-gram to be expanded
**Result**: $G'$: List of expansions for the input $n$-gram
if $g.getN() = 1$ then
    $G' \leftarrow []$
    foreach $synonym \in synonyms[word]$ do
        $transparency \leftarrow translationProbability(synonym, word)$
        if $transparency \geq wordTransparencyThreshold$ then
            $G'.add(< [synonym], transparency >)$
        end
    end
    return $G'$
else
    $G' \leftarrow getExpanedNGrams(g.subNGram(1, N))$
    $word \leftarrow g[0]$
    foreach $synonym \in synonyms[word]$ do
        $transparency \leftarrow translationProbability(synonym, word)$
        if $transparency \geq wordTransparencyThreshold$ then
            foreach $< transparency', g' > \in G'$ do
                if $(transparency' \times transparency) \geq$
                $ngramTransparencyThreshold$ then
                    $G'.add($
                    $< [synonym\ g'], transparency' \times transparency >)$
                end
            end
        end
    end
end
```
Algorithm 1: Function to return all valid expansions of an n-gram

Since we have made use of expanded n-grams, we have defined the *n-gram transparency* to specify how bold an n-gram has occurred in the document. For example, if an n-gram has occurred in the document, its transparency is equal to one, while its corresponding expanded n-gram, which has not occurred in the document, but it has been added to the feature set in the expansion phase as an expanded n-gram, has a lower transparency. This transparency score is calculated by multiplying the translation probabilities of synonyms constituting the expanded n-gram. To be mentioned, the translation probabilities are obtained from a monolingual dictionary that we have constructed from a parallel corpus using MOSES [1].

Our second approach is proposed in order to avoid defining multiple feature types for detecting plagiarism cases with different levels of obfuscation. We have introduced a similarity function for measuring the similarity of two features in the feature comparison phase, instead of comparing the exact string value of the features. In this way, adding other types of feature objects to the feature set would be redundant; and the word n-grams feature will cover all contextual word n-grams, sorted n-grams, stemmed word n-grams, named entity n-grams and expanded n-grams. In this approach all pos-

[1] http://www.statmt.org/moses/

sible pairs of n-grams have to be compared against each other. The details about the similarity function explained in Section 2.2.

Strategies for feature set space reduction The documents being text aligned may be too long, and they may contain lots of information which will not necessarily help in finding related regions of two documents. Considering the whole content of the documents for text alignment may make the feature set space too large and may lead to inefficiency in terms of time in the next step which is *"common feature detection"*. Besides, it would add noise to our features and affect the precision. Moreover if we want to apply the similarity score function approach it is really needed to have small set of features for each document in order to make it feasible to compute the similarity score between all possible pairs of features. To overcome these issues, we suggest several strategies for reducing the number of feature objects taken into account as useful features for the text alignment task. In general, applying these techniques should improve the precision, while it may decrease the recall.

- **Stopword removal**
  Stopwords may be useful in detecting some types of plagiarism but in many other cases they could simply be omitted from the text, having no effect on recall. For example for no obfuscation there is no need to use stopwords since, common word n-gram features are enough to detect the plagiarized passages; while for random obfuscation or summary obfuscation, stopwords may help, because the common features of type word n-gram are too sparse.
- **IDF based word filtering**
  In this technique, n-grams with low IDFs are removed. The reason behind this decision is that a low value of IDF indicates that the n-gram is occurred in many documents. Hence these n-grams may be detected as common features with a high probability, while they are not necessarily signs of plagiarism or any kind of relation between two regions of texts. We have omitted 20% of the documents' tokens with this strategy.
- **Feature object filtering based on features distributions in the suspicious document**
  If a feature object is distributed evenly in the suspicious document, we can conclude that this feature is somehow related to the main content of the document. For example in a document about *Barak Obama*, his name would frequently occur anywhere in the document. Hence even if it is detected as a common feature, it can hardly be a sign of plagiarism, and if it is, it would not help in specifying the plagiarized region in the suspicious document. According to this fact, we have introduced a *dispersion measure* for the feature objects. This measure shows how the n-gram is distributed in the document. The more uniformly the n-gram is distributed in the document, the higher its dispersion score will be. If it is higher than the specified threshold, for a feature, the feature would be removed from the feature set. The dispersion score for n-gram features are computed as in Equation 3.

$$dispersion(f_i, d) = \sum_{p_j} p(f_i, p_j) \times \log p(f_i, p_j), \tag{3}$$

where p_j is the jth paragraph of the document, and $p(f_i, p_j)$ is computed as $c(f_i, p_j)/c(f_i, d)$. $c(f_i, p_j)$ and $c(f_i, d)$ are the occurrence count of feature f_i in the jth paragraph of the document and the whole document respectively.
It is noteworthy that this strategy is applied only on suspicious documents.

2.2 Detecting Common Features

This step is called "finding seeds" in the literature of this research [4]. In this step features of the suspicious document are searched in the source document using their string value. Then a set of feature pairs is created. Each feature pair contains information about a feature in the suspicious document and its corresponding feature in the source document, such as feature offsets and feature values. Also a weight is assigned to each feature pair. When applying the exact comparison method, this weight is calculated based on feature transparencies. When applying the similarity score function based method, the similarity score assigned to each pair of feature is set as the pair weight. In the similarity function score method feature objects from the two documents should be compared the similarity function should applied on all pairs of features. The similarity function can take into account several aspects of features. We have mapped the comparison procedure of two word n-grams to the bipartite graph-matching problem. Words from the first n-gram are considered as the nodes of one part of the graph, and words from the second n-gram are considered as the nodes of the other part of the graph. The links between nodes of this bipartite graph is weighted according to the similarity score of their end point nodes. In our experiments we set this similarity score to be the translation probabilities obtained from a monolingual dictionary. For computing the overall similarity score between the two n-grams, a maximum weight-matching algorithm is applied on the graph, and then the normalized sum of the scores of the selected links is calculated as the similarity score of the n-grams. A threshold value, $SimTh$, is set, and if the similarity score of the n-grams is higher than that, these two features are considered as common features.

2.3 Clustering Common Features

Given a list of feature pairs as the common features, our mission is to detect regions of texts from the two documents where feature objects are shared densely enough. Taking into account the occurrence positions of feature objects in the pairs, they are mapped into a two dimensional space. The first dimension of this space represents the length of the first document and the second dimension represents the length of the second document. For each matched pair of feature objects, considering the occurrence positions of the features related to the source and suspicious documents, all possible pairs of offsets are computed as points in the 2D space. Finally a density based clustering algorithm, which is able to exclude the outlier points is applied to detect pairs of related regions. In our experiment we have used DBSCAN [2] as the density based clustering method. After clustering feature pairs based on their offsets in the source and suspicious documents, we have to determine the edges of the regions in both documents. This is done considering the offsets of the beginning and end of the sentences containing common features. We have decided to do this to make sure the regions of text that are selected as the plagiarized or source passages are meaningful units, which means a sentence may be included in the passage completely or not at all.

2.4 Post-processing of Detected Pairs

This post-processing step is aimed at handling the overlapping detected pairs. If this phase is done properly, it can affect granularity and precision significantly. Currently we have just designed a rule-based scenario for this part of the procedure, which is as follows: Detected cases are added sequentially to the final list. Upon adding a new case it is checked if it overlaps with the existing cases. If an overlap is found, considering the lengths of the cases and the length of the overlapping region, they are either merged or one of them is dropped. This step may further be improved, employing a classifier to accept or discard the detected cases.

3 Experiments

In order to design the experiments several parameters related to each phase of the procedure had to be taken into account. For the feature extraction phase, we need to specify the exact types of the features. In our experiments, word bi-grams to 5-grams are used. It should be mentioned that it is not reasonable to exploit uni-grams due to the fact that the number of common features increases significantly if uni-grams are used as feature objects; also the distribution of feature objects would be near uniform, which makes the clustering impossible. For the dispersion measure we have introduced in section 2.1, $DispTh$, we have set it to 0.4. Furthermore in our case DBSCAN, clustering parameters should be configured. For DBSCAN, we have to specify ϵ, the neighborhood diameter, and k, the minimum number of points in the neighborhood. Although automatic or semi-automatic algorithms are suggested for setting these parameters [2] , they are not actually appropriate in the application of text alignment. That is because the goal of the algorithms for determining the value of these thresholds automatically is a dynamic parameter setting regarding the characteristics of different data. For example, the neighborhood diameter should have different values in two cases of high-density area of points, or sparse collection of points. This is based on the assumption that the scale of the distances between points is not important, while this assumption is not valid when our goal is to merge parts of text in a document for text alignment. Hence, we have set these parameters heuristically. Finally we have set the value of k to 6 and ϵ to 380. Furthermore, applying the semantic similarity function technique, for common feature detection, at first it is needed to specify how the similarity function measures the similarity between two words. In our experiments, we have used a monolingual probabilistic dictionary, considering only the top 10 synonyms for each word to assign a similarity score to each pair of words. The monolingual probabilistic dictionary we have used is constructed from a parallel corpus of English and a temporary language. We have employed MOSES to create two bi-lingual dictionaries from the parallel corpus. Then we have constructed the EN-EN dictionary applying the probability rule in Equation 4 on these two dictionaries.

$$p(a|b) = \sum_x p(a|x)p(x|b). \tag{4}$$

In this equation a and b are English words, and x is a word from the temporary language. $p(a|x)$ and $p(x|b)$ are obtained from the bi-lingual dictionaries created by MOSES.

Furthermore the similarity score threshold, $SimTh$, for common features should be set, considering the probability distributions of synonyms of the words. The larger this threshold is, the higher the precision is, but we will encounter a lower recall. In our experiments, we have set the value of this threshold to 0.02. At first, we have tested our methods on PAN 2014 and 2013 text alignment corpora; then we have investigated the performance of our proposed method for different types of plagiarism cases when different feature sets are used.

3.1 Experiments On PAN 2014 datasets

The evaluation results of our method using simple word 3-grams and 4-grams, on PAN 2014 test Corpus II and III, achieved on TIRA platform, [3] is shown in Table 1.

Table 1. Applying the expanded n-gram based method setting the value of N to 3 and 4 on PAN 2014 test corpora

corpus	**precision**	**recall**	**granularity**	**plagdet score**
PAN 2014 test corpus II	0.77330	0.61163	1.02245	0.67220
PAN 2014 test corpus III	0.72686	0.67422	1.01169	0.69372

On these test corpora we have achieved a low recall, the reason behind this fact is that the values 3 and 4 considered for n-grams in this run, are not small enough to be able to detect common regions of text in summary and random obfuscation plagiarism cases. Since on corpus II we have achieved a higher precision, while we have obtained higher recall on corpus III, we can conclude that the plagiarism cases of corpus III were more close to the no-obfuscation or random obfuscation types of plagiarism.

Table 2 shows the result of applying the similariy score function based method on PAN 2014 test corpus III. We were not able to run this method successfully on corpus II, since it took too much time. We should investigate why this has happened upon we have access to the datasets.

Table 2. Applying our proposed similarity function based method using word 2-gram and 3-gram on PAN 2014 test corpus III

precision	**recall**	**granularity**	**plagdet score**
0.54833	0.84779	1.00455	0.66377

3.2 Experiments on PAN 2013 Test Dataset

Since we didn't have access to the PAN 2014 datasets we have run the experiments to investigate the performance of exploiting different feature sets per plagiarism type, on PAN 2013 test corpus [5].

Tables 3 and 4 illustrate plagdet scores obtained on PAN 2013 test dataset per plagiarism type when the similarity score function based method is applied.

Table 3. Plagdet score of the similarity score function based approach exploiting word n-grams and (n+1)-grams

plagiarism type	**N=2**	**N=3**	**N=4**	**N=5**
no obfuscation	0.2984	0.7865	0.8735	**0.8838**
random obfuscation	0.5427	**0.5754**	0.2627	0.1396
translation obfuscation	0.3442	**0.7751**	0.7540	0.6268
summary obfuscation	**0.4467**	0.3548	0.1367	0.0732

Table 4. Plagdet score of the similarity score function based approach exploiting word n-grams

plagiarism type	**N=2**	**N=3**	**N=4**	**N=5**
no obfuscation	0.3181	0.81321	0.8802	**0.8859**
random obfuscation	**0.5615**	0.5428	0.2359	0.1176
translation obfuscation	0.3764	**0.7517**	0.7220	0.5900
summary obfuscation	**0.4633**	0.3147	0.1217	0.0506

In Figure 1, the effect of increasing the value of n is illustrated when exploited features are simple word n and $n + 1$ grams. In Figure 2, the effect of increasing the value of n is illustrated when all types of features including the expanded n-grams are exploited.

As can be seen, setting n to 2 and 3 rather than larger values, led to better performance for random-obfuscation and summary obfuscation plagiarism types, while 3 and 4 grams work better for translation obfuscation, and the larger the value of n is, we obtain higher score for no-obfuscation plagiarism. Taking these results into account, we can conclude using a dynamic feature selection strategy, to select the features and setting the value of N, depending on the type of the plagiarism case can improve the results significantly.

4 Conclusion and Future Works

In this paper, we have discussed our approaches for solving the text alignment problem regarding the PAN challenge. We applied two different techniques for detecting the common features. Our first solution for common feature extraction was to compare

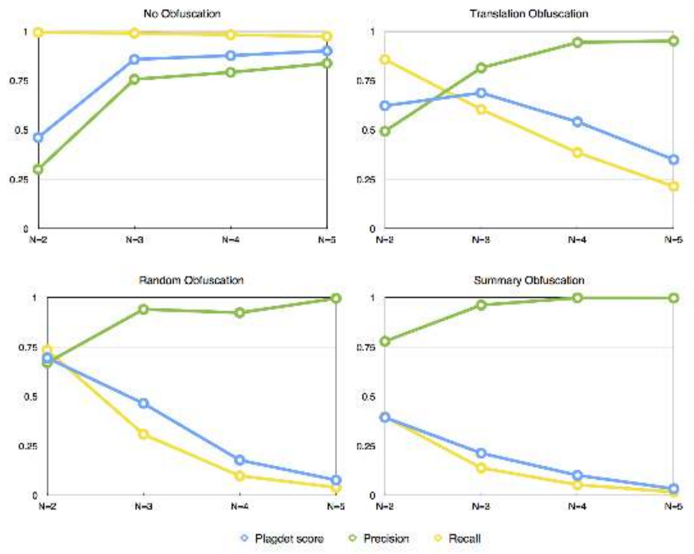

Figure 1. Effect of increasing the value of n when exploited features are n and $n+1$ word grams

features from the two documents in a fuzzy manner using a similarity function. The advantage of this technique is that, it reduces the size of the feature space since it is not needed to define many different types of features in this way. The disadvantage of this solution, is its long execution time specially for long documents. The second solution was to add the expanded word n-grams to the features set to be able to detect the semantic relations between documents and then, use a Hash-based comparison on features so as to detect the intersections between source and suspicious documents. The advantage of this solution is that it can be implemented highly efficient, and its disadvantage is that so as to be able to detect different levels of plagiarism with this technique we have to define many more different types of features. Moreover during this experience we explored how different combinations of feature sets affect the text aligners performance when applied on different type of plagiarism cases.

However, the results we obtained were not as good as other participants of the PAN text alignment track both in terms of performance and efficiency, there exist several potentials so as to improve it. As a future work, it should be worked on the efficiency of the similarity score function based method. Additionally, The experiments indicate that for all types of n-gram features, customizing the value of n per plagiarism type may lead to better results. Therefore, adding dynamic feature selection step to the process of text alignment is a good idea. Furthermore learning algorithms can be used for setting some parameters like the dispersion threshold or the similarity score threshold.

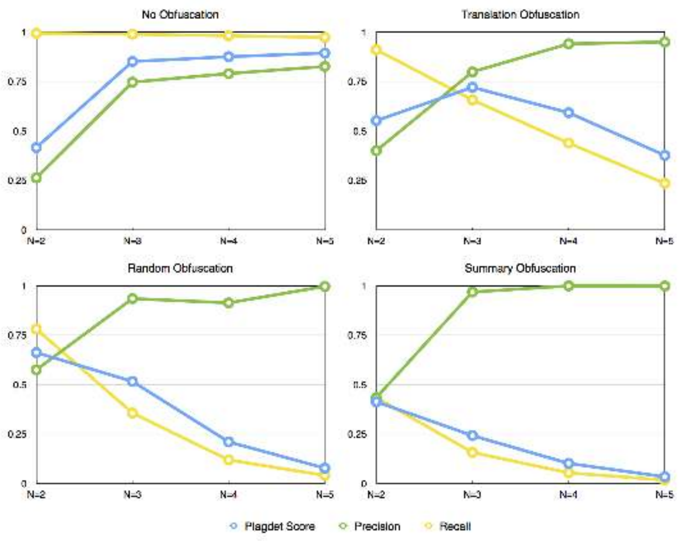

Figure 2. Effect of increasing the value of n is when all feature types are exploited

References

1. Peter F Brown, Jennifer C Lai, and Robert L Mercer. Aligning sentences in parallel corpora. In *Proceedings of the 29th annual meeting on Association for Computational Linguistics*, pages 169–176. Association for Computational Linguistics, 1991.
2. Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, volume 96, pages 226–231, 1996.
3. Tim Gollub, Martin Potthast, Anna Beyer, Matthias Busse, Francisco Rangel, Paolo Rosso, Efstathios Stamatatos, and Benno Stein. Recent trends in digital text forensics and its evaluation. In Pamela Forner, Henning Müller, Roberto Paredes, Paolo Rosso, and Benno Stein, editors, *Information Access Evaluation meets Multilinguality, Multimodality, and Visualization. 4th International Conference of the CLEF Initiative (CLEF 13)*, 2013.
4. Martin Potthast, Tim Gollub, Matthias Hagen, Johannes Kiesel, Maximilian Michel, Arnd Oberländer, Martin Tippmann, Alberto Barrón-Cedeno, Parth Gupta, Paolo Rosso, et al. Overview of the 4th international competition on plagiarism detection. In *CLEF (Online Working Notes/Labs/Workshop)*, 2012.
5. Martin Potthast, Matthias Hagen, Tim Gollub, Martin Tippmann, Johannes Kiesel, Paolo Rosso, Efstathios Stamatatos, and Benno Stein. Overview of the 5th international competition on plagiarism detection. In Pamela Forner, Roberto Navigli, and Dan Tufis, editors, *Working Notes Papers of the CLEF 2013 Evaluation Labs*, 2013.
6. Efstathios Stamatatos. Plagiarism detection using stopword n-grams. *Journal of the American Society for Information Science and Technology*, 62(12):2512–2527, 2011.
7. Diego Antonio Rodríguez Torrejón and José Manuel Martín Ramos. Coremo system (contextual reference monotony) a fast, low cost and high performance plagiarism analyzer system: Lab report for pan at clef 2010. In *CLEF (Online Working Notes/Labs/Workshop)*, 2010.