Scalable and Language-Independent Embedding-based Approach for Plagiarism Detection Considering Obfuscation Type: No Training Phase

Erfaneh Gharavi
Data & Signal Processing Lab,
Faculty of new Sciences and Technologies,
University of Tehran

e.gharavi@ut.ac.ir

Hadi Veisi
Data & Signal Processing Lab,
Faculty of new Sciences and Technologies,
University of Tehran

h.veisi@ut.ac.ir

Paolo Rosso
PRHLT Research Center,
Universitat Politècnica de València

prosso@dsic.upv.es

ABSTRACT

The efficiency and scalability of plagiarism detection systems have become a major challenge due to the vast amount of available textual data in several languages over the Internet. Plagiarism occurs in different levels of obfuscation, ranging from the exact copy of original materials to text summarization. Consequently, designed algorithms to detect plagiarism should be robust to the diverse languages and different type of obfuscation in plagiarism cases. In this paper, we employ text embedding vectors to compare similarity among documents to detect plagiarism. Word vectors are combined by a simple aggregation function to represent a text document. This representation comprises semantic and syntactic information of the text and leads to efficient text alignment among suspicious and original documents. By comparing representations of sentences in source and suspicious documents, pair sentences with the highest similarity are considered as the candidates or seeds of plagiarism cases. To filter and merge these seeds, a set of parameters, including Jaccard similarity and merging threshold, are tuned by two different approaches: offline tuning and online tuning. The offline method, which is used as the benchmark, regulates a unique set of parameters for all types of plagiarism by several trials on the training corpus. Experiments show improvements in performance by considering obfuscation type during threshold tuning. In this regard, our proposed online approach uses two statistical methods to filter outlier candidates automatically by their scale of obfuscation. By employing the online tuning approach, no distinct training dataset is required to train the system. We applied our proposed method on available datasets in English, Persian and Arabic languages on the text alignment task to evaluate the robustness of the proposed methods from the language perspective as well. As our experimental results confirm, our efficient approach can achieve considerable performance on the different datasets in various languages. Our online threshold tuning approach without any training datasets works as well as, or even in some cases better than, the training-base method.

Keywords
Text Alignment; Language-Independent Plagiarism Detection; Word Embedding; Text Representation; Obfuscation Type

1. INTRODUCTION

The vast amount of available text materials over the internet has become the main source of information acquisition and retrieval in the current era. Using them as one's own without explicitly acknowledging the materials' originator is considered plagiarism, which is an illegitimate immoral act (Clough, 2003). This act has intensified through the fast and easy access to the contents. The overwhelming amount of text production every day makes manual approaches to detect plagiarism based on the reviewer's knowledge impractical. Various algorithms in natural language processing (NLP) specifically in text similarity detection, come along to detect plagiarism automatically. Plagiarism occurred with different types of obfuscation, ranging from exact copy to idea plagiarism and text summarization. Considering obfuscation type in detecting plagiarism improves the performance of the plagiarism detection systems (Sanchez-Perez et al., 2014). In some categorizations, plagiarism detection is divided into two main categories: external plagiarism detection, and intrinsic plagiarism detection (Potthast et al., 2009). External plagiarism detection examines all the available source documents and tries to detect the potential source text for a given suspicious documents. On the other hand, intrinsic plagiarism detection tries to find the anomaly of the text writing style to discover parts of the text which are not written by the same author of the whole document (Hoad & Zobel, 2003). External plagiarism also varies at different levels, like modification in the vocabulary, alterations in syntactic-structure, and transformation in semantic representation of the text. Text alignment is an external plagiarism detection task that tries to find the source of text in a suspicious document.

One of the main concerns in NLP tasks including text alignment is representing text in a computer-readable format to be used in other processing steps. Traditional approaches define each term as one feature and represent a document as a vector of terms occurrence. Several numbers of unique words in a corpus make processing of this sparse vector time consuming and inefficient (Bengio et al., 2003). In this paper, we use word representation introduced by (Firth, 1957) to represent texts and evaluate the similarity between them to detect plagiarism. In this so-called word embedding representation, a word is described by its surrounding neighbors. This representation comprises semantic and syntactic information for each word. Describing text by embedding representation resulted in a higher performance in several NLP tasks with almost no hand engineering feature extraction and preprocessing (Collobert et al., 2011). Beyond words, a representation of phrase, sentence, and documents exposed a new challenging task in this field. A number of algorithms were proposed of text representation, ranging from simple summation (Mitchell & Lapata, 2010) to complex neural network based composition functions (Socher, 2014).

In this paper, we improved our method for text similarity detection applied in the text alignment shared-task (Gharavi et al., 2016). This new enhanced approach is employed on different datasets from various languages and obfuscation type perspectives. In this task, a pair of documents is given to identify the possible plagiarism among them. Due to the efficiency issues, we apply word vector averaging to represent a sentence. Obfuscation type was also taken into account while comparing the text parts. In the proposed method, we employed two approaches for filtering the detected plagiarism cases. The first approach tuned the required threshold by lots of trials over a training dataset. We also set the threshold by considering obfuscation type, which improves the total performance of the system. In the second approach, two methods are employed to remove outliers in the filtering phase, which make threshold tuning among the training data set inessential. These methods automatically tune the threshold with respect to obfuscation type. The main advantages of our approach among others are its simplicity and its fast candidate selection. We accelerated the process by transforming a n-gram-by-n-gram comparison, so-called string-matching approach, to a numerical one. Synthetic changes in sentences, including alteration in word order which resulted in the same representation of sentences can be detected conveniently with our approach. This method can easily identify transformations in vocabulary, including adding or omitting words, which would be indistinguishable by string matching approaches. Furthermore, we proposed a new approach for threshold tuning that omits the training phase. The system could be run in real time and with almost no training data. We proved the scalability of our obfuscation-aware language-independent approach to any new corpus without training data, for different languages on the diverse datasets.

The rest of this paper is organized as follows: Section 2 presents the related works for the text alignment task. Section 3 illustrates the text embedding representation. Section 4 defines the proposed method and Section 5 illustrates the experiments, datasets, evaluation metrics and results. Finally, we illuminate the characteristics of our method in the conclusion section.

2. RELATED WORK

The challenging task of automated plagiarism detection has captured researchers' attention (Chong et al., 2010; Clough, 2003). This indicates an increase in the demand for such systems as well as the need to improve their speed and efficiency. Here we explain some of the methods applied to text alignment shared task.

Previous methods mostly benefit from a method called string matching scheme in order to detect copied texts (Potthast et al., 2014) Beyond an exact copy of the text, in some cases detecting plagiarism is tough even for humans. When it comes to computers, such methods like string matching lack the flexibility to encounter different type of plagiarism caused by modifications in the syntactic and semantic structure of the text. Text alignment was considered as a sequence alignment in bioinformatics in the previous methods (Potthast et al., 2014). As illustrated in Figure 1, a number of steps usually are taken into account to build a text alignment system: (1) seeding, (2) extension, and (3) filtering. In text alignment methods, two source and suspicious documents are compared to each other by using some heuristic seeds. These seeds are the exact matches or created matches by changing the text in different ways. After defining a lot of seeds, the method tries to extend these seeds to align text passages. In the filtering step, cases that do not meet certain criteria, i.e. overlapping cases or tremendously short passages, are discarded (Potthast et al., 2014).

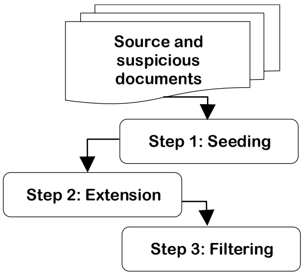

Figure 1: General steps toward plagiarism detection

We explain various employed methods for each of three phases of a plagiarism detection system here. Sorted word 4-grams and unsorted stop word 8-grams are employed as seeds in (Suchomel et al., 2013). These seeds are merged if the gap between them is below some pre-defined thresholds. Then, two seeds with no more than 4000 characters apart are merged. They reject cases under 300 characters in the filtering phase. Sorted word 3-grams and two kinds of sorted word 1-skip-grams are used as seeds by Rodríguez Torrejón and Martín Ramos (Rodríguez Torrejón et al., 2014). In Leilei et al. (Leilei et al., 2012) a pair of sentences is considered as seeds if they exceed a similarity threshold. For the filtering phase, detected cases that have their modified Jaccard co-efficient among them under a specific threshold are removed in Leilei et al. (Leilei et al., 2013). Shrestha and Solorio (Shrestha & Solorio, 2013) use 5-grams that hold at least one name entity, called name entity 5-grams along with stopword 8-grams as seeds. In (Shrestha et al., 2014) also a method for inexact n-gram matching that have their Jaccard similarity above a predefined threshold is presented.

Palkovskii and Belov (Palkovskii & Belov, 2013) considered word n-grams, stop word n-grams, named entity n-grams, frequent word n-grams, stemmed n-grams, sorted n-grams, and skip-n-grams as seeds. They employed clustering for unsupervised merging. In (Palkovskii & Belov, 2014) the authors tried to apply four corresponding parameter sets based on four kinds of obfuscation. These obfuscation types include no obfuscation, random obfuscation, summaries, and "undefined". Alvi et al. (Alvi et al., 2014) use character 20-grams and Rabin-Karp algorithm for string matching. They discard alignments of less than 200 character length in the source document and less than 100 character length in the suspicious document. Skip word 2-grams ranging from 1 to 4 are used by Gross and Modaresi (Gross & Modaresi, 2014) to match the exact cases. For merging the seeds, they applied agglomerative single-linkage clustering, where pairs of seeds are merged based on a distance measure. Alignments of fewer than 15 words length are discarded.

Glinos in (Glinos, 2014) use top 30 most frequent words longer than 5 characters and match the exact cases. A gene sequence alignment algorithm is employed to detect match cases. Instead of gene sequences, they tailored the application for a pair of text. Clusters of related words to the topics considered to detect highly obfuscated plagiarism. This clustering takes place by a set of hand-crafted rules. They employed logistic regression to train a classifier-based on lexical similarity features using the training dataset of the evaluation corpus. However, the incorporation of the prediction into the text alignment remains unclear.

In (Sanchez-Perez et al., 2014) each sentence is represented as a term-frequency inverse document frequency (TF-IDF) weighted vector. These vectors are compared by cosine similarity and Dice coefficient measures. The threshold for passing these two measures was 0.33. They applied an approach that relates to divisive clustering. All subsequent seeds are merged using a gap threshold, and then they divided the merged text until they exceed a similarity threshold. They also distinguished between summary cases and all the other type of obfuscation and employed two parameter settings for their approach at the same time. Afterwards, the decision on outputs was made on the basis of the length difference between suspicious passage and source passage. For example, they considered a candidate seed as a summary case if the source passage is more than 3 times longer than the suspicious passage.

For Persian text alignment task, the source document is indexed into the trie structure in (Talebpour at al., 2016). Then the suspicious document is analyzed word by word against trie structure to find potential sources. In another work in Persian language, n-grams are considered as seeds (Minaei & Niknam, 2016) and short detected cases are discarded in filtering phase. Momtaz and her colleagues (Momtaz et al., 2016) turn the sentences from source and suspicious documents into graphs and tried to find similar graphs. In (Esteki & Esfahani, 2016) sentences are classified as similar and non-similar by Support Vector Machine (SVM) classifier. In this work, the Levenshtein distance, the Jaccard coefficient, and the Longest Common Subsequence (LCS) extracted from pairs of sentences are considered as the features. TF-IDF weighting is used in (Mashhadirajab & Shamsfard, 2016) to create sentence vectors from source and suspicious documents to find seeds. They use SVM neural network to predict obfuscation type to tune the required parameters.

The distributed representation of words (word embeddings) is used in (Ferrero, Besacier, Schwab, & Agnès, 2017) for cross-language textual similarity detection. In (Agarwal, Ramampiaro, Langseth, & Ruocco, 2018) an informative semantic representation of each sentence is created by (1) using CNN to extract the local region information in form of important n-grams from the sentence, and (2) applying RNN to capture the long-term dependency information within paraphrase detection. The proposed approach in (Ehsan, Shakery, & Tompa, 2018) has two main steps: the first step tries to find candidate plagiarised fragments and focuses on high recall, followed by a more precise similarity analysis based on dynamic text alignment that will filter the results by finding alignments between the identified fragments. The novel text representation scheme proposed in (Sánchez-Vega et al., 2019) gathers both content and style characteristics of texts, represented employing character-level features. Authors in (Al-Suhaiqi, Hazaa, & Albared, 2018) evaluates five different monolingual plagiarism detection methods namely i) n-grams similarity, ii) longest common subsequence, iii) dice coefficient, iv) fingerprint-based jaccard similarity and v) fingerprint-based containment similarity. In addition, three machine learning classifiers are used for Arabic-English cross-language plagiarism detection.

This paper is inspired by our previous work (E. Gharavi, Veisi, Bijari, & Zahirnia, 2018). The approach is extended by adding two methods for threshold tuning without using the training dataset and also applied in English and Arabic corpora to prove the scalability and language independency of our method.

3. TEXT REPRESENTATION

Primary approaches in NLP represented text as a sparse vector with the length corresponding to the size of the lexicon. This vector has values in existing words positioning and initializes other elements with zeros. These values varies in binary format or number of word occurrence in the document. Considering words as a single feature makes natural language processing tasks unable to identify similarities among synonym words. Subsequently, the idea of representing a word by its neighbors was introduced by Firth (Firth, 1957). We explain this representation and the idea of combing them to construct context representation at the phrase, sentence or document levels in the following subsections.

3.1 Word Embedding

Distributed word representation, generally known as word-embedding, is used to solve the problems of high dimensionality and sparsity in the language model. In this representation, each word is described by the surrounding context which contains semantic and syntactic information about the word. A language modeling task is employed to construct such representation. Using this representation, synonym words have similar vectors (Socher, 2014).

Distributed representation learning is introduced by Hinton for the first time (Hinton, 1986) and is developed as a language modeling concept by Bengio (Bengio et al., 2003). Collobert (Collobert et al., 2011) showed that distributed representation of words with almost no engineered features can be shared by several NLP tasks resulting in equal or more accuracy than state-of-the-art methods. Finally, authors in (Mikolov et al., 2013) indicated that this kind of representation not only encompasses a huge part of syntactic and semantic rules, but also the relationship between words can be modeled by vectors' offset. This offset can also present the plurality, syntactic label (noun, verb, etc.) and the semantic feature (pet, animal, car, etc.) of a word.

Context-based methods such as skip-grams or continuous bag of words are usually employed to learn word representation (Mikolov et al., 2013; Pennington et al., 2014). In these approaches, three-layer feed-forward neural networks are trained for the language modeling task. Skip-gram uses the one-on representation of words in a limited window size as an input and try to predict the middle word in the context. Another version of this network, continuous bag of words, is used to predict the context considering the middle word.

3.2 Phrase/Sentences/Paragraph Embedding

So far, we have explained how to represent words in a way that the representation contains semantic and syntactic information. However, we usually need a rich representation to demonstrate a phrase, sentence or paragraph or a document. Many algorithms are represented for this purpose. They usually introduce a composition function to combine the above-mentioned word embedding. Some of them are reviewed in the following.

Socher introduces variations of unsupervised (Socher et al., 2011) and supervised (Socher, 2014) recursive neural networks (RNNs). This method uses the idea of the hierarchical structure of the text and encodes two word-vectors into one vector by auto-encoder networks. Socher also presents many variations of these deep combination functions such as RNN (Socher et al., 2010) and Matrix-Vector Recursive Neural Networks (MV-RNN) (Socher et al., 2012). Long short-term memory (LSTM) which captures long-distance dependency among text is also employed to model sentences (Tai et al., 2015). Convolution neural networks (CNN) also work well in tasks such as sentiment analysis (Kalchbrenner et al., 2014). Paragraph Vector (Le & Mikolov, 2014) is an unsupervised algorithm that learns representation for variable-length pieces of texts. The algorithm assumes each paragraph as a vector, which is updated during the language modeling task. Paragraph vector outperforms other methods, such as bag-of-words models, for many applications. Feature cluster-based vector space model (FC-VSM) (Qimin et al., 2015) used the text feature clusters co-occurrence matrix to represent document for clustering purpose. Livermore et al in (Livermore et al., 2018), use a representation learning based on the combination of Topic Modeling and the Citation Networks to retrieve the latent relevance structure of the documents.

Word vector characteristics make them capable of being composed by basic mathematical methods (Mitchell & Lapata, 2010). These methods include summation, multiplication and averaging. The results prove the ability of these methods to represent the semantic information of each sentence. In this paper, we employed the averaging approach to take advantages of both rich semantic representation and efficiency due to its simplicity in the calculation.

4. PROPOSED METHOD

In this paper, we propose a sentence by sentence comparison approach to align source and suspicious documents. As shown in Figure 2, we followed a two-phase method for this aim: in the first phase, two sentences are nominated as candidate seeds for plagiarism and then, in the second phase, all the candidates are filtered to identify correct plagiarism cases.

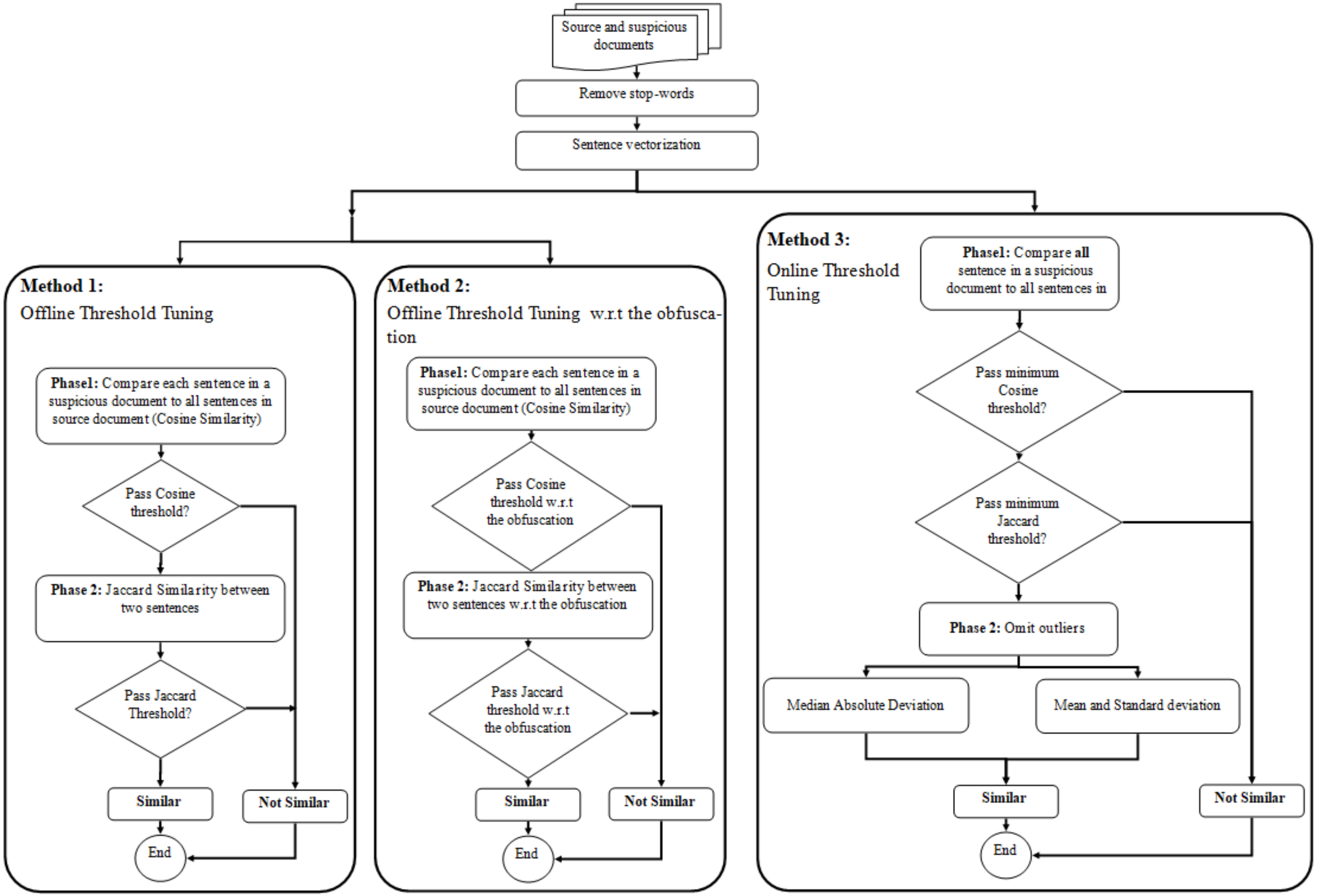
Source and suspicious documents
Remove stop-words
Sentence vectorization
Method 1:
Offline Threshold Tuning
Phasel: Compare each sentence in a suspicious document to all sentences in source document (Cosine Similarity)
Pass Cosine threshold?
Phase 2: Jaccard Similarity between two sentences
Pass Jaccard Threshold?
Similar
Not Similar
End
Method 2:
Offline Threshold Tuning w.r.t the obfusca-tion
Phasel: Compare each sentence in a suspicious document to all sentences in source document (Cosine Similarity)
Pass Cosine threshold w.r.t the obfuscation
Phase 2: Jaccard Similarity between two sentences w.r.t the obfuscation
Pass Jaccard threshold w.r.t the obfuscation
Similar
Not Similar
End
Method 3:
Online Threshold Tuning
Phasel: Compare all sentence in a suspicious document to all sentences in
Pass minimum Cosine threshold?
Pass minimum Jaccard threshold?
Phase 2: Omit outliers
Median Absolute Deviation
Mean and Standard deviation
Similar
Not Similar
End

Figure 2: Proposed plagiarism detection methods (α: cosine similarity threshold, β: Jaccard similarity threshold, γ: gap merging threshold)

We used pre-trained GoogleNews[1] vectors for English word vectors and we trained word vectors for Persian language using the word2vec algorithm (Mikolov, Corrado, et al., 2013). Available pre-trained Arabic word vectors[2] are used for the Arabic language (Bojanowski et al., 2016). For capturing the whole sentence's idea, only substantial words remain and stop words are removed from the sentences.

In the first phase, the semantic similarity of two sentences is evaluated. As mentioned in Section 3.1, word embedding embraces semantic and syntactic information for each word. Since system efficiency is one of the main concerns in real plagiarism detection applications, we chose the word vector's average as the composition function to construct sentence embedding. In this regard, we average all significant word embedding vectors. Equation 1 illustrates sentence vector calculation:

$$S_i = \frac{\sum_{j=1}^{n} w_{ij}}{n}, \qquad (1)$$

where S_i is the vector representation for sentence i and w_{ij} is the word vector for jth word of sentence i and n is the number of words in that sentence. Th resulting vector represents the semantic information of each sentence. Each dimension depicted a feature to describe that sentence. All the sentences in the source and suspicious documents are represented by this approach.

After the feature extraction phase, we compare each sentence representation vector in the suspicious document to all the sentence representation vectors in the source document. Here, cosine similarity is used as a comparison metric. This metric is described in Equation 2.

$$CosineSimilairy = \frac{S_{src} \cdot S_{sus}}{\|S_{src}\| \|S_{sus}\|} = \frac{\sum_{i=1}^{K} S_{sus_i} S_{sus_i}}{\sqrt{\sum_{i=1}^{K} S_{sus_i}^{2}} \sqrt{\sum_{i=1}^{K} S_{sus_i}^{2}}} \qquad (2)$$

In which, S_{src} is the sentence vector of a sentence from a source document and $S_{sus}$is the sentence vector of a sentence from a suspicious document and K denotes the vectors' dimension.

For each sentence in the suspicious document, we nominated one of the sentences in the source document with the highest cosine similarity greater than a predefined threshold, as a plagiarism candidate. This candidate, which is retrieved by Equation (3), is the most semantically similar sentence to the suspicious one. The semantic similarity of the sentences comes from the fact that each word of them is represented by a word embedding vector that contains semantic information of the word implicitly.

$$plag\_candidate(S_i) = \underset{j}{\operatorname{argmax}} \; (cosine\_sim(S_i, S_j)) \qquad (3)$$

Where S_i is a sentence in the suspicious document and S_j is a sentence in the source document. Since this is a numerical calculation and is not including any time-consuming string matching, this candidate is retrieved in real time. To approve this candidate, i.e. accept these two sentences as a plagiarism case, we also used the Jaccard similarity measure to assess the lexical similarity of two sentences. Jaccard similarity score is calculated as in Equation (4):

$$Jaccard\_Similarity_{word} \; (SW_1, SW_2) = \frac{|SW_1 \cap SW_2|}{|SW_1 \cup SW_2|}, \qquad (4)$$

where SW_1 is the set of unique words in the first sentence and SW_2 is the set of unique words in the second sentence. This similarity measure calculates the number of common words and normalizes it by the total number of unique words in both sentences. We considered stopwords here without considering word frequency. In some cases, although the sentences passed the semantic similarity measure and became a candidate, the Jaccard similarity among them at word level was trivial despite their likeliness. For example, let us consider two sentences below:

A="These studies have argued the facts."
B="This study arguing the fact."

Although these two sentences should be considered similar, the Jaccard similarity at word level is very low, Jaccard_Sim$_{\text{word}}$ (A, B) = 0.1. Due to efficiency matter, complicated preprocessing, such as the transformation of word form to their roots (arguing, argued➔ argue) was not taken into account. To obviate this variation, the similarity of two sentences is also calculated at character level. In this regard, instead of words, n-grams are considered as sentence elements. Equation (5) depicts the Jaccard similarity at character level,

$$Jaccard\_Similarity_{character} \; (SC_1, SC_2) = \frac{|SC_1 \cap SC_2|}{|SC_1 \cup SC_2|}, \qquad (5)$$

where SC_1 is the set of unique n-grams in the first sentence and SC_2 is the set of unique n-grams in the second sentence. Here, we chose 3-grams as segments. So two aforementioned sentences segmented as follow:

A="The se stu die s hav e arg ued the fac ts."
B="Thi s stu dy arg uin g the fac t"

[1] https://github.com/mmihaltz/word2vec-GoogleNews-vectors

[2] https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md

Now the Jaccard similarity at character level is 0.31 which is satisfactory in the filtering phase.
Finally, two candidate sentences which pass the semantic similarity threshold in the first phase and the lexical similarity threshold in the second phase either at the word *or* character level are considered to be plagiarism. We applied different approaches to fine-tune thresholds. This can be achieved by various trial over training corpus or decided locally with respect to the level of obfuscation. These methods are illustrated in the next section.

4.1 Parameter Tuning

For detecting plagiarism cases, three parameters are needed to be optimized, either by tuning over the training corpus (offline) or tuning regarding obfuscation type (online). We define these parameters as below:

- α = Cosine similarity threshold to retrieve initial candidate.
- β = Jaccard similarity to approved plagiarism cases.
- γ = the maximum gap, number of characters, between two sentences to decide whether to merge two detected cases or not.

Jaccard similarity at character level is tuned regarding Jaccard similarity at the word level. A ε=0.2 is added to the word level Jaccard similarity to define character level threshold.
We proposed three different methods to tune these parameters.

4.1.1 Method 1: Offline threshold tuning

Two sentences are considered to be plagiarism if they pass the cosine similarity threshold (α). The second threshold (β), Jaccard similarity threshold, filters the selected sentences to assure lexical similarity. Two sentences are merged if they have fewer characters between them than the specific threshold (γ). We called this approach *offline threshold tuning* since all the parameters are tuned by several trials on the training corpus. The combination of these parameters that achieve the highest-plagdet is selected using training data. In each round of experiment, we change one parameter while keeping the others constant. Sets of evaluated parameters are illustrated in Table 1.

Table 1: Range of threshold parameters

Parameters	Range
α	[0.0-1.0]
β	[0.0-1.0]
γ	{0, 100, 500, 800, 2000, 4000, 5000}

During the experiment, we realized that α parameter affects recall evaluation measure and β parameter has an effect on precision. These perceptions seem logical since α threshold determine seeds and its variation change number of selected seeds and β threshold filter the selected seeds so its fluctuation affects precision.

4.1.2 Method 2: Offline threshold tuning with respect to the obfuscation type

During the parameter optimization phase, we realized that changing these parameters with respect to the obfuscation type leads to better results. For example, if we increase the cosine similarity thresholds in plagiarism cases with no obfuscation from 0.3 to 0.5, i.e. exact copy, the precision increase from 0.8775 to 0.9598 without any or considerable loss for other cases in recall measure. Therefore, for evaluation purpose, we fine-tune all these parameters with respect to the obfuscation type in different languages. We call this approach *offline threshold tuning w.r.t the obfuscation type.* Also decreasing this threshold for summary cases leads to higher recall and better system performance.

4.1.3 Method 3: Automatic language-independent parameter tuning with respect to the obfuscation type, online tuning

In the previous section, we discussed how obfuscation type-aware parameter tuning increased the system's performance. But in real applications, we never know the obfuscation type before we detect it. Designing a system to detect type of obfuscation in advance and select the parameters with respect to the output of this system is one approach. In this regard, we tried to build a neural network classifier to detect type of obfuscation by considering similarity thresholds as its input. This system did not achieve reasonable accuracy to be applicable. Due to this, we decided to define the parameters locally in each document. Here, we assume that each document pair has a unique kind of obfuscation if there are any plagiarism cases in it.
A minimum threshold is assigned to retrieve all possible candidates in both evaluation phases. Then, γ is assigned in the next phase with respect to the level of similarity. After candidate retrieval, we applied two statistical methods using mean and median on Jaccard similarity values to omit outliers. The decision on the threshold is taken without considering the language or corpus. We called this approach *online threshold tuning* since the minimum of all parameters is assigned locally depending on the level of obfuscation in plagiarism cases. This approach deprived lots of trials on the training corpus. Statistical methods to detect outliers are described in the following sections.

a. Outlier detection method: standard deviation

In the first method, the mean of the reported Jaccard similarity values is calculated. For detecting outliers, the standard deviation related to those values is estimated. Assume J = {J_1, J_2, …, J_N} as all Jaccard similarity values between each two sentence pairs greater than the minimum threshold. Equation (6) and (7) show the calculation of mean and standard deviation of these values,

$$\bar{J} = \frac{1}{N}\sum_{i=1}^{N} J_i \qquad (6)$$

$$std = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(J_i - \bar{J})^2} \qquad (7)$$

where N is the number of similarity values for plagiarism candidates. Since most of the sentences with same obfuscation type have similarity measure in the same range, we consider values less than $(\bar{J} - std)$ as outliers. Equation (8) define the β threshold calculation.

$$\beta = \bar{J} - std \qquad (8)$$

In outlier detection methods, the method should consider the similarity measure higher than $(\bar{J} + std)$ also as the outliers, but in this case the greater the similarity, the higher the plagiarism probability. Therefore, we consider them as plagiarism cases.

b. Outlier detection method: median absolute deviation

Standard deviation works better on data with a normal distribution. Therefore, we implement another statistical method called Median Absolute Deviation (MAD) to find outliers. In this method, first similarity values are sorted to find the median. Then we calculate the absolute difference of the median and all similarity values. Afterward, a new array of Jaccard similarity values is sorted and the new median is selected for these numbers. We subtract this number from the first median and define the minimum threshold for considering two sentences as plagiarism. Again, assume J = {J_1, J_2, …, J_N} as all Jaccard similarity values between each two sentence pairs greater than the minimum threshold. Equation (9) describes the MAD method.

$$\beta = MAD = median(|J_i - median(J)|) \qquad (9)$$

We consider cases with similarity less than MAD as the outliers.

Considering mean and median values we decide which number of characters between two sentences (γ) is acceptable to merge two plagiarism cases for the document based on these numbers. For example, mean and median around [0.95-1.00] is common in no-obfuscation type and in this case decreasing γ leads to a more precise candidate selection.

5. EXPERIMENTS

In this section, the datasets and evaluation metrics are first presented. Then, the results of evaluations are given.

5.1 Dataset

For text alignment tasks, evaluation corpora are compiled based on different obfuscation levels. In general, these corpora are comprised of the following plagiarism types (Potthast et al., 2013):

- *No-obfuscation*: The exact same version of the original text is in the suspicious document.
- *Random obfuscation:* Adding, removing and exchanging vocabularies by other synonyms and shuffling words in the text are included in this obfuscation type. This kind of obfuscation can easily be created by artificial approaches. Creating this so-called artificial cases is cheaper and easier than the upcoming once.
- *Cyclic translation:* A piece of text is translated from original language to a target one and then translated back to the original language. This back and forth cyclic translation keeps the semantic meaning while representing another version of the text.
- *Summary obfuscation:* In this case, a text is summarized into an abstract version while the whole idea remains the same. In this type of obfuscation, the lexical and syntactic similarity is very restricted.
- *Simulated obfuscation:* Samples are manually constructed using human resources and crowdsourcing. People are asked to read and paraphrase the text and rewrite it with other phrases.

In our evaluations, to check the language-independency, we have evaluated our method for three languages, English, Persian, and Arabic. PAN-PC-2013 text alignment corpus[3] which is also used in PAN-PC-2014 is employed as the English dataset (Potthast et al., 2013) for evaluations on the English language. The dataset used in PersianPlagDet2016[4] consider as the corpus for text alignment in Persian language (Asghari et al., 2016).
To the best of our knowledge, there is no available text alignment corpus for the Arabic language, so we converted the available dataset[5] for external plagiarism detection to a corpus that fit for the text alignment task. For this purpose, a pair file was created according to the

[3] http://pan.webis.de/data.html

[4] http://ictrc.ac.ir/plagdet/

[5] http://misc-umc.org/AraPlagDet/?i=1

available information and an XML file containing all source files related to this document was separated based on the distinct source files. These processes prepared the data for the text alignment task.
The statistics of these corpora for all languages is illustrated in Table 2. All of these datasets contains four types of plagiarism. Table 3 gives details of the statistics from different types of obfuscation aspects for these three corpora.

Table 2: Corpus statistics (number of documents) for three languages

# of documents	English	Persian	Arabic
Source file	489	1563	567
Suspicious file	398	1525	607

Table 3: Corpus statistics based on the plagiarism types for three languages

# of cases	English	Persian	Arabic
No-plagiarism	90	783	169
No-obfuscation	108	208	490
Random/Translated	199	1611	615
Simulated/Summary	121	147	163

We considered 10 percent of available data for parameter tuning in the first method and did the evaluations on the rest of the data. For other methods, all documents are used in evaluations, since we do not have the training phase.

5.2 Evaluation Metrics

Three evaluation measures are employed to assess text alignment approaches. These measures include precision, recall, and granularity, which are combined into the plagdet score (Potthast et al., 2010). Precision and recall are calculated as in the following equations.

$$Prec(S,R) = \frac{1}{|R|}\sum_{r\in R}\frac{|\cup_{s\in S}(S \sqcap r)|}{|r|} \qquad (10)$$

$$Rec(S,R) = \frac{1}{|S|}\sum_{s\in S}\frac{|\cup_{r\in R}(S \sqcap r)|}{|s|} \qquad (11)$$

$$\text{Where } S \sqcap r = \begin{cases} s \cap r \; if \; r \; detects \; s, \\ \emptyset \;\; otherwise. \end{cases}$$

s is a plagiarism case *and S* is the set of plagiarism cases in the corpus, r associates an allegedly plagiarized passage and *R* is the set of detected plagiarism cases for the suspicious documents.

F₁ is the harmonic mean of precision and recall. This measure is calculated by Equation (12).

$$F1 = \frac{2 \times Prec \;\times Rec}{Prec + Rec} \qquad (12)$$

Granularity is defined to address overlapping or multiple detections for one plagiarism case and is calculated as in Equation (13). This metric evaluates the detector system for its integrity of detecting each plagiarism case.

$$gran(S,R) = \frac{1}{|S_R|}\sum_{s\in S_R}|R_s| \qquad (13)$$

Where $S_R \subseteq S$ are cases detected by detections in R, and $R_s \subseteq R$ are detections of s. All these measures are combined into a single score, PalgDet, as in Equation (14):

$$plagdet(S,R) = \frac{F_1}{\log_2(1 + gran(S,R))} \qquad (14)$$

This measure is used to report the performance of a text alignment system.

5.3 Results and Discussions

In this section, all the results of the three methods applied to three languages including English, Persian and Arabic are presented.

5.3.1 *Method 1: Offline threshold tuning*

As mentioned in the proposed method section, two sentences are considered to be plagiarism if they pass the cosine (α) and Jaccard (β) similarity thresholds; and they are merged if the gap between them is not more than a specific threshold (γ). Table 4 shows the fine-tuned thresholds achieved by several trials on the training corpus.

Table 4: Best values for different parameters (α: cosine similarity threshold, β: Jaccard similarity threshold, γ: gap merging threshold)

Parameter	English	Persian	Arabic
α	0.3	0.3	0.3
β	0.38	0.25	0.3
γ	2000	2200	2200

We run the algorithms over the three training corpora and separately on different cases of plagiarism for each language. Table 5, Table 6 and

Table 7 present the results of our method on English, Persian and Arabic corpora by parameters set in Table 4, respectively. In these tables, the result of plagiarism detection over the total corpus is also given as *All Types*. We can see through the results that the best performance is achieved in no-obfuscation cases since two vectors for two same sentences are exactly the same. For the most difficult cases, the summary obfuscation type, the PlagDet is poor due to its high general threshold. In this type of obfuscation, although the system could not retrieve all the cases, still, the accuracy of the retrieved ones is proven by the highest precision.

Table 5: Plagiarism detection performance on the **English** corpus for different obfuscation types (α=0.3, β=0.38, γ=2000)

	Recall	Precision	Gran	PlagDet
All Types	0.7369	0.8655	1.0028	0.7944
No-Obfuscation	0.9167	0.8775	1.0000	0.8967
Random	0.6702	0.8235	1.0000	0.7390
Translated	0.6857	0.8995	1.0000	0.7782
Summary	0.3248	0.9412	1.0000	0.4830

Table 6: Plagiarism detection performance on the **Persian** corpus for different obfuscation types (α=0.3, β=0.25, γ=2200)

	Recall	Precision	Gran	PlagDet
All Types	0.9080	0.9430	1.0000	0.9248
No-Obfuscation	0.9860	0.9566	1.0000	0.9711
Random	0.9159	0.9492	1.0000	0.9319
Simulated	0.7264	0.9572	1.0000	0.8260

Table 7: Plagiarism detection performance on the **Arabic** corpus for different obfuscation types (α=0.3, β=0.3, γ=2200)

	Recall	Precision	Gran	PlagDet
All Types	0.7978	0.6295	1.0017	0.7029
No-Obfuscation	0.9768	0.7062	1.0021	0.8185
Artificial	0.6121	0.5263	1.0000	0.5660
Simulated	0.9141	0.8465	1.0000	0.8790

5.3.2 *Method 2: Offline threshold tuning with respect to the obfuscation type*

For evaluation purpose, we also fine-tune these three parameters with respect to the obfuscation type in the different languages. Parameters that achieved the best result are reported in Table 8. As it is shown, the values of these parameters are highly dependent on the obfuscation type. Also, it can be seen that the values are language dependent.

Table 8: Best values for different parameters depend on obfuscation type

		No-obfuscation	Random/ Translated	Summary/ Simulated
English	α	0.5	0.3	0.3
	β	0.45	0.32	0.25
	γ	600	1000	4000
Persian	α	0.6	0.3	0.3
	β	0.45	0.27	0.23
	γ	400	1000	3000
Arabic	α	0.3	0.3	0.3
	β	0.4	0.35	0.3
	γ	1000	2200	2200

Table 9, Table 10 and Table 11 present the results of applying different parameter sets considering obfuscation type. We also collected all cases together to report the performance of the entire corpus, indicated as *All Types* in the tables.

Table 9: Plagiarism detection performance on the **English** corpus (α, β, γ selected depends on obfuscation type)

	Recall	Precision	Gran	PlagDet
All Types	0.8049	0.8291	1	0.8168
No-Obfuscation	0.9254	0.9398	1	0.9325
Random	0.7256	0.7930	1	0.7578
Translated	0.7721	0.8011	1	0.7863
Summary	0.5694	0.6503	1	0.6072

Table 10: Plagiarism detection performance on the **Persian** corpus (α, β, γ selected depends on obfuscation type)

	Recall	Precision	Gran	PlagDet
All Types	0.9071	0.9555	1	0.9307
No-Obfuscation	0.9821	0.9793	1	0.9807
Random	0.9157	0.9494	1	0.9323
Simulated	0.7784	0.9236	1	0.8448

Table 11: Plagiarism detection performance on the **Arabic** corpus (α, β, γ selected depends on obfuscation type)

	Recall	Precision	Gran	PlagDet
All Types	0.7894	0.6421	1.0009	0.7077
No-Obfuscation	0.9870	0.7083	1.0021	0.8236
Artificial	0.6121	0.5263	1.0000	0.5660
Simulated	0.9133	0.8522	1.0000	0.8817

As can be seen, the performance is improved in the obfuscation type-aware threshold tuning approach for all the cases. Comparing the results, we can note a significant improvement in summary cases in English corpus, almost 12% in PlagDet. This result is considered as the 3rd best-reported systems' performance on this type of obfuscation for this dataset (Potthast et al., 2014). This improvement was achieved by a substantial change in recall rate. Comparing β values in Table 4 and Table 8 we can notice that this progress is due to the decrease in Jaccard similarity threshold from *0.38* to *0.25.* With this new threshold, summary obfuscation cases that usually have lower similarity can be detected. This is the same for summary cases in the Persian language where the threshold is decreased from 0.25 to 0.23. Despite summary cases, the performance of plagiarism cases without any obfuscation, i.e. exact copy, improved by increasing the thresholds. As we can see, PlagDet is increased by 4% in the English language and 1% in other languages. In this case, precision improvement resulted in the PlagDet increase. By intensifying the threshold, none of the irrelevant cases have passed. Therefore, the number of false positive cases drops significantly.

For all the three languages, plagiarism cases in random or translated cases have not fluctuated by different thresholds. The fine-tune threshold was the same or very close to the best of these cases.

5.3.3 Method 3: Automatic language-independent parameter tuning with respect to obfuscation type, online threshold tuning

Results in the previous section proved the effectiveness of obfuscation type-aware parameter tuning on plagiarism detection performance. Since the obfuscation type is not defined before detecting the case, this method is not applicable. Here, the parameters are tuned in each document separately, assuming the existence of just one unique kind of obfuscation in each document pair.
All possible candidates were retrieved in both evaluation phases by a assigning minimum threshold of 0.3 to α and β. And we indicated γ in the next phase with respect to the obfuscation type. To have a deeper look at the selection of the value for γ the distribution of β values with respect to the different plagiarism types are depicted using histogram charts. As it is shown in Figure 3, for no obfuscation plagiarism cases, the value of Jaccard similarity is 1 for almost all the cases. The frequency of cases with Jaccard similarity in range (0.5-0.8) is higher for Random/Translated cases. And for summary/Simulated cases most of the cases have Jaccard similarity lower than 0.5. We use these distributions to regulate γ. For this purpose, mean and median range [0-1] is divided into three segments. Table 12 states the range of β parameters and corresponding γ thresholds.

Table 12: γ threshold with .respect .to the obfuscation type

Type of obfuscation	Range of β	γ
None	[0.8-1]	800
Random/Translated	(0.5-0.8)	2000
Summary/Simulated	[0.3-0.5]	4000

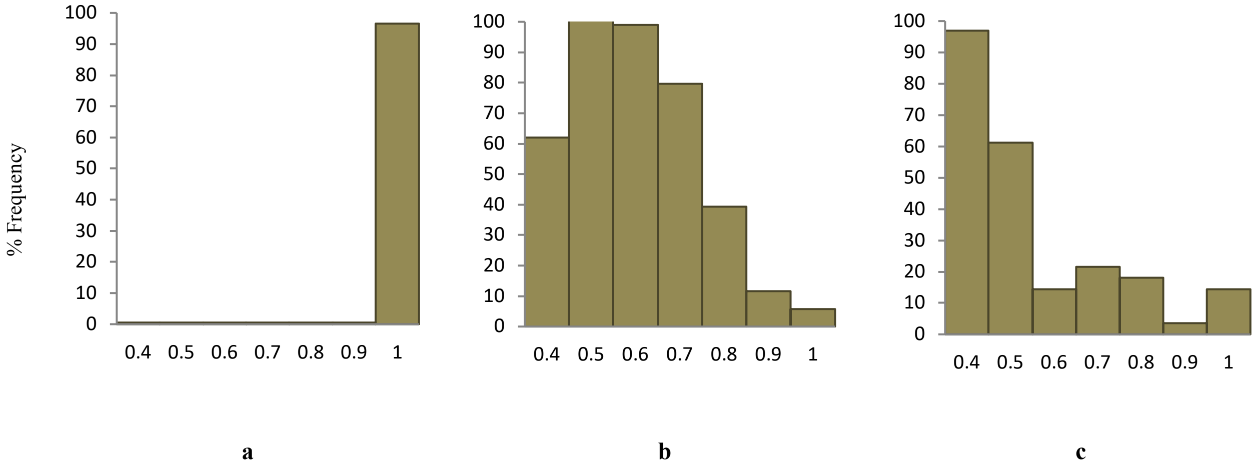

Figure 3: Jaccard similarity distribution, β, over different types of obfuscation,
a: No-obfuscation cases, **b**: Random/Translated cases, **c**: Summary/Simulated cases

As mentioned in section 4.1.3 two statistical approaches were used to detect outliers which their results are reported in the following sub-sections. The comparison of these two methods with the previous one is reported in Table 19.

a. Outlier detection method: standard deviation

Table 13,

Table **14** and Table 15 illustrate detection performance with online threshold tuning method using standard deviation as outlier detection on English, Persian and Arabic datasets, respectively.

Table 13: Plagiarism detection performance on the **English** corpus (α=0.3, β, γ tuned online by std)

	Recall	Precision	Gran	PlagDet
All Types	0.7723	0.8056	1.0027	0.7871
No-Obfuscation	0.8845	0.9007	1.0000	0.8926
Random	0.7325	0.7280	1.0100	0.7251
Translated	0.7473	0.7712	1.0000	0.7591
Summary	0.4534	0.8118	1.0000	0.5818

Table 14: Plagiarism detection performance on the **Persian** corpus (α=0.3, β, γ tuned online by std)

	Recall	Precision	Gran	PlagDet
All Types	0.8965	0.9501	1	0.9225
No-Obfuscation	0.9735	0.9864	1	0.9799
Random	0.9042	0.9553	1	0.9291
Simulated	0.7190	0.9583	1	0.8216

Table 15: Plagiarism detection performance on the **Arabic** corpus (α=0.3, β, γ tuned online by std)

	Recall	Precision	Gran	PlagDet
All Types	0.7938	0.6403	1.0000	0.7089
No-Obfuscation	0.9867	0.7065	1.0000	0.8234
Artificial	0.6096	0.5317	1.0000	0.5680
Simulated	0.9093	0.8480	1.0000	0.8775

The results show that this method works almost as well as the manually setting the threshold in all cases. Even in summary cases for English language, automatic threshold tuning outperforms initial method by 10 percent.

b. Outlier detection method: median absolute deviation

The MAD method is employed as another outlier detection method. We consider cases with similarity less than the MAD threshold which is calculated by Equation (9) as the outliers. Table 16, Table 17 and Table 18 present detection performance with this method.

Table 16: Plagiarism detection performance on the **English** corpus (α=0.3, β, γ tuned online by MAD)

	Recall	Precision	Gran	PlagDet
All Types	0.7671	0.8336	1	0.7990
No-Obfuscation	0.8839	0.9379	1	0.9101
Random	0.7290	0.7510	1	0.7399
Translated	0.7349	0.8007	1	0.7664
Summary	0.4534	0.8118	1	0.5818

Table 17: Plagiarism detection performance on the **Persian** corpus (α=0.3, β, γ tuned online by MAD)

	Recall	Precision	Gran	PlagDet
All Types	0.8836	0.9540	1	0.9175
No-Obfuscation	0.9723	0.9869	1	0.9795
Random	0.8897	0.9594	1	0.9233
Simulated	0.7074	0.9660	1	0.8167

Table 18: Plagiarism detection performance on the **Arabic** corpus (α=0.3, β, γ tuned online by MAD)

	Recall	Precision	Gran	PlagDet
All Types	0.7975	0.6350	1.0008	0.7066
No-Obfuscation	0.9867	0.7050	1.0021	0.8212
Artificial	0.6178	0.5229	1.0000	0.5664
Simulated	0.9069	0.8479	1.0000	0.8764

For English language, median noise filtering method works better than the standard deviation. But for Persian and Arabic languages the standard deviation noise detection method outperform median one. This might be due to the different distribution of plagiarism cases among different languages.

Table 19: Summary of plagiarism detection performance on English, Persian & Arabic corpora, respectively

		Recall	Precision	Gran	PlagDet
English	Method 1	0.7369	0.8655	1.0028	0.7944
	Method 2	0.8049	0.8291	1.0000	0.8168
	Method 3: standard deviation	0.7723	0.8056	1.0027	0.7871
	Method 3: MAD	0.7671	0.8336	1.0000	**0.7990**
Persian	Method 1	0.9080	0.9430	1	**0.9248**
	Method 2	0.9071	0.9555	1	0.9307
	Method 3: standard deviation	0.8965	0.9501	1	0.9225
	Method 3: MAD	0.8836	0.9540	1	0.9175
Arabic	Method 1	0.7978	0.6295	1.0017	0.7029
	Method 2	0.7894	0.6421	1.0009	0.7077
	Method 3: standard deviation	0.7938	0.6403	1.0000	**0.7089**
	Method 3: MAD	0.7975	0.6350	1.0008	0.7066

Table 19 summarizes the results of all aforementioned methods in English, Persian, and Arabic, respectively. As it can be seen in this table, the method with local threshold tuning, online, works as well as the tuning threshold with lots of trials on training datasets, offline. In the English language, the online method which applied MAD as outlier detector outperforms offline parameter tuning on the corpus. For the Arabic language, the online threshold tuning outperforms the other methods. The online method performs as well as the offline one for the Persian language with insignificant differences, less than 0.25% in comparison with the mean outlier detection method. The results prove the feasibility of our novel approach to different language and different types of obfuscation. Being independent of any training data makes our model applicable to any new datasets without required pre-training.

6. CONCLUSIONS

In this paper, we proposed an efficient, language-independent and scalable approach for the text alignment task. This approach leads to a robust model to detect plagiarism cases on the various monolingual datasets in different languages. In our approach, sentence-by-sentence comparison is applied to find analogous sentences. Three methods were presented to detect similar sentences and the results were reported for each one on the different datasets.

In the first method, we tuned the thresholds of similarity metrics by several trials on the training corpus. The main advantages of this method among others are its simplicity and its fast candidate selection. In the presented approach, comparison transformed from a word-by-word or n-gram-by-n-gram representation of text data, so-called string matching, to a numerical one. In this regard, the calculation of similarity could be executed in a much faster and more convenient way. Two sentence pairs with no obfuscation have the exact same vector, which can be addressed with high accuracy. Synthetic changes in sentences, including alteration in word order which resulted in the same vectors can also be detected conveniently with our approach. This method can easily identify transformations in vocabulary, include adding or omitting words, which would be indistinguishable by string matching approaches. This is due to the insignificant effect of these kinds of modifications on sentence representation vectors.

In plagiarism cases where the words substituted by their synonyms, the resemblance of synonym word vectors leads to similar representation for two sentences. In addition, time-consuming use of lexicon for synonym retrieval will be eliminated. In these cases, two sentences have lower similarity measure value so the threshold should be low. Therefore, obfuscation-aware threshold-tuning was proposed. In the second method, obfuscation type is taken into account during the threshold tuning phase. We realized that considering the obfuscation type in threshold tuning for similarity detection task improves performance. In the third method, parameters were regulated regarding all sentence similarity values in a document pair. The assumption is that there is one type of plagiarism in each document pair. In this method, all sentence pairs in all languages pass a fixed minimum threshold to be considered as a plagiarism candidate. Then two statistical methods are applied to remove outliers from candidates. One method finds the outliers outside the mean minus variance range and the other used the median to find the outliers. In the online threshold tuning approach, we modified threshold values based on obfuscation type.

Results showed that online parameter tuning regarding the obfuscation type resulted in almost the same and, in some cases, better performance than the training-based approaches. The online tuning outperforms the offline corpus-based tuning approach in English and Arabic corpora. Raising the threshold value leads to higher precision in plagiarism cases with no obfuscation cases and lowering it leads to higher recall in the summary and simulated cases.

In comparison to the other approaches in the text alignment task, the highest PlagDet on the English dataset for no-obfuscation cases was 93.25%, achieved by our second approach. This is the fourth best-reported performance on this dataset. The best PlagDet reported for this type of obfuscation is 96.2%. For random and translated cases, the performance of the system is moderate due to the problems with finding the sentence boundaries. For the summary cases, we achieved the third best result on the English dataset. The best PlagDet reported for this type of obfuscation is 62.35% and we achieved 58.18%. On the entire corpus, the best-reported result, 87.81%, is 8% higher than our best result, 79.9%. Due to the impossibility of detecting the type of obfuscation beforehand, we did not take the result of the second method into account for comparison. For the Persian and Arabic datasets, the specific test datasets were not publicly available to compare with. The performance evaluation of our initial approach compared to the others for the Persian language is reported in Asghari et al. (Asghari et al., 2016). Our approach has the best runtime on the Persian dataset, and achieved a comparable PlagDet to the winning team on the Persian plagiarism detection contest.

As illustrated and proved in the result section, the proposed methods work in different languages and on the various corpora. Online decision-making processes assist us to decide on selecting a local threshold with respect to the obfuscation type without any requirement to the training data. As our experimental results confirm, our efficient, language-independent, scalable approach can be applied to the various corpora in the different languages.

Acknowledgement:

The work of Paolo Rosso was partially funded by the Spanish MICINN under the research project MISMIS-FAKEnHATE on Misinformation and Miscommunication in social media: FAKE news and HATE speech (PGC2018-096212-B-C31).

Competing interests

The authors declare that they have no competing interests.

REFERENCES

Agarwal, B., Ramampiaro, H., Langseth, H., & Ruocco, M. (2018). A deep network model for paraphrase detection in short text messages. *Information Processing & Management*, *54*(6), 922–937.

Al-Suhaiqi, M., Hazaa, M. A. S., & Albared, M. (2018). Arabic English Cross-Lingual Plagiarism Detection Based on Keyphrases Extraction, Monolingual and Machine Learning Approach. *Asian Journal of Research in Computer Science*, 1–12.

Alvi, F., Stevenson, M., & Clough, P. D. (2014). Hashing and Merging Heuristics for Text Reuse Detection. *CLEF (Working Notes)*, 939–946.

Asghari, H., Mohtaj, S., Fatemi, O., Faili, H., Rosso, P., & Potthast, M. (2016). Algorithms and corpora for Persian plagiarism detection. *CEUR Workshop Proceedings*, *1737*, 135–144.

Bengio, Y., Ducharme, R., Vincent, P., & Janvin, C. (2003). A Neural Probabilistic Language Model. *The Journal of Machine Learning Research*, *3*, 1137–1155. https://doi.org/10.1162/153244303322533223

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2016). Enriching Word Vectors with Subword Information. *ArXiv Preprint ArXiv:1607.04606*.

Chong, M., Specia, L., & Mitkov, R. (2010). Using Natural Language Processing for Automatic Detection of Plagiarism. *Language*. Retrieved from http://clg.wlv.ac.uk/papers/show_paper.php?ID=272

Clough, P. (2003). Old and new challenges in automatic plagiarism detection. *National Plagiarism Advisory Service*, (February), 14. Retrieved from http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Old+and+new+challenges+in+automatic+plagiarism+detection#0

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*, *12*, 2493–2537. https://doi.org/10.1.1.231.4614

Ehsan, N., Shakery, A., & Tompa, F. W. (2018). Cross-lingual text alignment for fine-grained plagiarism detection. *Journal of Information Science*, 0165551518787696.

Esteki, F., & Esfahani, F. S. (2016). A plagiarism detection approach based on SVM for Persian texts. *CEUR Workshop Proceedings*, *1737*, 149–153.

Ferrero, J., Besacier, L., Schwab, D., & Agnès, F. (2017). Using Word Embedding for Cross-Language Plagiarism Detection. *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. https://doi.org/10.18653/v1/E17-2066

Firth, J. R. (1957). A synopsis of linguistic theory, 1930-1955. *Studies in Linguistic Analysis*, pp. 1–32.

Gharavi, E., Veisi, H., Bijari, K., & Zahirnia, K. (2018). A Fast Multi-level Plagiarism Detection Method Based on Document Embedding Representation. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. https://doi.org/10.1007/978-3-319-73606-8_7

Gharavi, Erfaneh, Bijari, K., Veisi, H., & Zahirnia, K. (2016). *A Deep Learning Approach to Persian Plagiarism Detection*. Retrieved from https://pdfs.semanticscholar.org/b0a8/7335289264368a7ee804acc7715fc4799310.pdf

Glinos, D. G. (2014). A Hybrid Architecture for Plagiarism Detection. *CLEF (Working Notes)*, 958–965.

Gross, P., & Modaresi, P. (2014). Plagiarism Alignment Detection by Merging Context Seeds. *CLEF (Working Notes)*, 966–972.

Hinton, G. (1986). Learning distributed representations of concepts. *CSS*, pp. 1–12. https://doi.org/10.1109/69.917563

Hoad, T. C., & Zobel, J. (2003). Methods for identifying versioned and plagiarized documents. *Journal of the American Society for Information Science and Technology*, Vol. 54, pp. 203–215. https://doi.org/10.1002/asi.10170

Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014). A Convolutional Neural Network for Modelling Sentences. *Acl*, pp. 655–665.

https://doi.org/10.3115/v1/P14-1062

Le, Q. V., & Mikolov, T. (2014). *Distributed Representations of Sentences and Documents. 32.* https://doi.org/10.1145/2740908.2742760

Leilei, K., Haoliang, Q., Cuixia, D., Mingxing, W., & Zhongyuan, H. (2013). Approaches for source retrieval and text alignment of plagiarism detection: Notebook for PAN at CLEF 2013. *CEUR Workshop Proceedings*, *1179*.

Leilei, K., Haoliang, Q., Shuai, W., & Cuixia, D. (2012). Approaches for Candidate Document Retrieval and Detailed Comparison of Plagiarism Detection. *Notebook for PAN at CLEF 2012*. Retrieved from http://www.uni-weimar.de/medien/webis/research/events/pan-12/pan12-papers-final/pan12-plagiarism-detection/kong12-notebook.pdf

Livermore, M. A., Dadgostari, F., Guim, M., Beling, P., & Rockmore, D. (2018). Law Search as Prediction. *Virginia Public Law and Legal Theory Research Paper*, (2018–61).

Mashhadirajab, F., & Shamsfard, M. (2016). A Text Alignment Algorithm Based on Prediction of Obfuscation Types Using SVM Neural Network. *FIRE (Working Notes)*, 167–171.

Mikolov, T., Corrado, G., Chen, K., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *Proceedings of the International Conference on Learning Representations (ICLR 2013)*, 1–12. https://doi.org/10.1162/153244303322533223

Mikolov, T., Yih, W., & Zweig, G. (2013). Linguistic regularities in continuous space word representations. *Proceedings of NAACL-HLT*, (June), 746–751. Retrieved from http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Linguistic+Regularities+in+Continuous+Space+Word+Representations#0%5Cnhttps://www.aclweb.org/anthology/N/N13/N13-1090.pdf

Minaei, B., & Niknam, M. (2016). An n-gram based Method for nearly Copy Detection in Plagiarism Systems. *FIRE (Working Notes)*, 172–175.

Mitchell, J., & Lapata, M. (2010). Composition in Distributional Models of Semantics. *Cognitive Science*, *34*(8), 1388–1429. https://doi.org/10.1111/j.1551-6709.2010.01106.x

Momtaz, M., Bijari, K., Salehi, M., & Veisi, H. (2016). Graph-based Approach to Text Alignment for Plagiarism Detection in Persian Documents. *FIRE (Working Notes)*, 176–179.

Palkovskii, Y., & Belov, A. (2013). Using Hybrid Similarity Methods for Plagiarism Detection. *Notebook for PAN at CLEF 2013*.

Palkovskii, Y., & Belov, A. (2014). Developing High-Resolution Universal Multi-Type N-Gram Plagiarism Detector. *Working Notes Papers of the CLEF 2014 Evaluation Labs*, 984–989.

Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 1532–1543. https://doi.org/10.3115/v1/D14-1162

Potthast, M., Stein, B., Eiselt, A., Barrón-Cedeño, A., Rosso, P. (2009). Overview of the 1st International Competition on Plagiarism Detection. *SEPLN 09 Workshop on Uncovering Plagiarism, Authorship, and Social Software Misuse*, 1–9. Retrieved from http://ceur-ws.org/Vol-502

Potthast, M., Hagen, M., Beyer, A., Busse, M., Tippmann, M., Rosso, P., & Stein, B. (2014). Overview of the 6th International Competition on Plagiarism Detection. *Notebook for PAN at CLEF 2014*, 845–876.

Potthast, M., Hagen, M., Gollub, T., Tippmann, M., Kiesel, J., Rosso, P., … Stein, B. (2013). Overview of the 5th international competition on plagiarism detection. *CEUR Workshop Proceedings*, *1179*.

Potthast, M., Stein, B., Barrón-cedeño, A., & Rosso. (2010). An evaluation framework for plagiarism detection. *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)*, (August), 997–1005. Retrieved from http://dl.acm.org/citation.cfm?id=1944566.1944681

Rodríguez Torrejón, D., Martín Ramos, J. (2014). CoReMo 2.3 plagiarism detector text alignment module: Notebook for PAN at CLEF 2014. *CEUR Workshop Proceedings*, *1180*, 997–1003.

Sanchez-Perez, M. A., Sidorov, G., & Gelbukh, A. (2014). The winning approach to text alignment for text reuse detection at PAN 2014: Notebook for PAN at CLEF 2014. *CEUR Workshop Proceedings*, *1180*, 1004–1011.

Sánchez-Vega, F., Villatoro-Tello, E., Montes-y-Gómez, M., Rosso, P., Stamatatos, E., & Villaseñor-Pineda, L. (2019). Paraphrase plagiarism identification with character-level features. *Pattern Analysis and Applications*, *22*(2), 669–681.

Shrestha, P., Maharjan, S., & Solorio, T. (2014). Machine Translation Evaluation Metric for Text Alignment. *CLEF (Working Notes)*, 1012–1016.

Shrestha, P., & Solorio, T. (2013). Using a Variety of n-Grams for the Detection of Different Kinds of Plagiarism. *Notebook for PAN at CLEF*, *2013*.

Socher, R. (2014). Recursive Deep Learning for Natural Language Processing and Computer Vision. *PhD Thesis*, (August).

Socher, R., Huang, E., & Pennington, J. (2011). Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection. *Advances in Neural Information Processing Systems*, 801–809. Retrieved from http://machinelearning.wustl.edu/mlpapers/paper_files/NIPS2011_0538.pdf%5Cnhttps://papers.nips.cc/paper/4204-dynamic-pooling-and-unfolding-recursive-autoencoders-for-paraphrase-detection.pdf

Socher, R., Manning, C. D. C., & Ng, A. Y. A. (2010). Learning continuous phrase representations and syntactic parsing with recursive neural networks. *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*, 1–9. https://doi.org/10.1007/978-3-540-87479-9

Socher, R., Manning, C., Huval, B., & Ng, A. (2012). Semantic compositionality through recursive matrix-vector spaces. *EMNLP-CoNLL '12: Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and*

Computational Natural Language Learning, 1201–1211. https://doi.org/10.1162/153244303322533223

Suchomel, Š., Kasprzak, J., Brandejs, M., & others. (2013). Diverse queries and feature type selection for plagiarism discovery. *Notebook for PAN at CLEF 2013*.

Tai, K. S., Socher, R., & Manning, C. D. (2015). Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. *Proceedings of ACL*, 1556–1566. https://doi.org/10.1515/popets-2015-0023

Talebpour, A., Shirzadi, M., & Aminolroaya, Z. (2016). Plagiarism detection based on a novel trie-based approach. *CEUR Workshop Proceedings*, *1737*, 180–183.