A Two-Level Plagiarism Detection System for Arabic Documents

El Moatez Billah Nagoudi, Ahmed Khorsi, Hadda Cherroun, Didier Schwab

BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **XX**, No X
Sofia • 201X

A Two-Level Plagiarism Detection System for Arabic Documents

*El Moatez Billah Nagoudi *, Ahmed Khorsi **, Hadda Cherroun *, Didier Schwab****

* *Laboratoire d'Informatique et de Mathématique LIM, Amar Telidji University, Laghouat, Algeria.*
** *Al-Imam Mohammad Ibn Saud Islamic University, Saudi Arabia.*
*** *Laboratoire d'Informatique et de Grenoble GETALP, Univ. Grenoble Alpes, Grenoble, France.*
1 e.nagoudi@lagh-univ.dz 2 amakhorsi@imamu.edu.sa 3 h.cherroun@ lagh-univ.dz 4 didier.schwab@imag.fr

Abstract: *Measuring the amount of shared information between two documents is a key to address a number of Natural Language Processing (NLP) challenges such as Information Retrieval (IR), Semantic Textual Similarity (STS), Sentiment Analysis (SA) and Plagiarism Detection (PD). In this paper, we report a plagiarism detection system based on two layers of assessment: 1. fingerprinting which simply compares the documents fingerprints to detect the verbatim reproduction. 2. Word embedding which uses the semantic and syntactic properties of words to detect much more complicated reproductions. Moreover, Word Alignment (WA), Inverse Document Frequency (IDF) and Part-of-Speech (POS) weighting are applied on the examined documents to support the identification of words that are most descriptive in each textual unit. In the present work, we focused on Arabic documents and we evaluated the performance of the system on a data-set of holding three types of plagiarism: 1. Simple reproduction (copy and paste), 2. Word and phrase shuffling 3. Intelligent plagiarism including synonym substitution, diacritics insertion and paraphrasing. The results show a recall of 88% and a precision of 85%. Compared to the results obtained by the systems participating in the Arabic Plagiarism Detection Shared Task 2015, our system outperforms all of them with a plagiarism detection score (Plagdet) of 83%.*

Keywords: *Plagiarism Detection, Intelligent Plagiarism, Fingerprinting, Word Embedding, Arabic Language.*

1. Introduction

1.1. Plagiarism Detection

"*Plagiarism is the use of ideas, concepts, words, or structures without appropriately acknowledging the source to benefit in a setting where originality is expected*" [1]. The easy access to the vast amount of information on the net has shown to be an appealing opportunity for authors of diverse backgrounds to steel and claim others' works. In the last few years, the phenomenon has been reported to have spread over different areas including academia, literature, media and not to mention the industry [2]. In academia, for instance, a study conducted by Guibert and Michau [3], reported that about 35% of the students in Europe have re-used all

or a portion of a document to present it as their own work. McCabe [4] who studied a sample of more than 80.000 students in the U.S. and Canada between 2002 to 2005 showed that more than 25% of graduate and 38% of undergraduate students have at least copied or paraphrased sentences without citing the source.

Compared to formal languages (i.e. programs) plagiarism in natural language is relatively more difficult to identify because of the flexibility of morphology and syntax [8]. In addition, plagiarists use different ways to bypass the plagiarism detection systems. A plagiarism detection system task is then to uncover what the plagiarist did his best to hide using rewording, synonym substitution, paraphrasing, text manipulation, text translation and idea adoption [5].

From the perspective of the resources used to make the detection, there are two approaches: (1) Intrinsic, (2) Extrinsic [5]. The first one examines the linguistic features of a document against itself to spot the catching variations in styles, this technique is known as stylometry [11]. The extrinsic plagiarism, however, compares the suspicious document with a source collection of documents [9]. One could say the first technique tries to find the differences while the second tries to spot similarities.

1.2. Arabic Language

As the language we study in the present work brings a number of linguistic challenges we extend this section with a brief introduction to this language. The Arabic language is a Semitic language with rich and complex morphology compared to the Indo-European languages [6]. It is spoken by more than 330 million people as a native language and it is the fourth most used language on the Internet[1]. Arabic is written from right to left and it has 28 alphabet letters. In Arabic text, letters are attached and they change shape in accordance with their position in the word. In the other hand, diacritic marks may optionally be present, consequently, for a word with k letters, we can have at least 2^k different representations [7]. Another issue is the fact that some letters are frequently used interchangeably, such as (ى ، ي), (ة ، ه) and (ا ، أ ، إ ، آ) [34].

1.3. Our contribution

In this paper, we present a *Two-Level Arabic Plagiarism Detection* system (2L-APD), built around the extrinsic plagiarism detection approach. The proposed 2L-APD system is based on two modules (levels), (1) Fingerprinting detection module, (2) Word embedding detection module. The first one is designed to detect literal reproduction of texts. The word embedding detection module tries to discover synonym substitution and paraphrasing if any.

The rest of this paper is organised as follows, in the next section we provide a quick overview of Arabic plagiarism detection published works. Section 3 introduces some background on fingerprinting and word embedding models. Section 4 describes the architecture of our plagiarism detection system 2L-APD. In

[1] http://www.internetworldstats.com/stats7.htm

Section 5, we report the test results and compare it to the results of similar systems. Finally, our conclusions and some future research directions are drawn in Section 6.

2. Arabic Plagiarism Detection Systems

While we focus on Arabic language the interested reader may refer to a number of surveys on the subject of plagiarism detection in general and on other languages [27], [9], [5] and [2]. In the context of Arabic language, several plagiarism detection systems are proposed. For instance, Alzahrani and Salim [28] have introduced a statement-based plagiarism detection system for Arabic (FS-APD) using fuzzy-set information retrieval model [18]. The degree of similarity between two statements is computed and compared to a fixed threshold value to judge whether are similar or not. This approach led to perform well on verbatim reproductions. To address the rewording, they have proposed another system named fuzzy semantic-based string similarity for extrinsic plagiarism detection (SFS-APD) [30]. This uses a shingling algorithm, Arabic WordNet lexical database [31] and Jaccard coefficient for retrieving a list of candidate documents. The suspicious document is then compared sentence by sentence with the candidate documents to compute the fuzzy degree of similarity.

Meni [6] proposed a plagiarism detection tool for Arabic documents (Aplag). Aplag is based on heuristics to compare suspect documents at different hierarchical levels to avoid unnecessary comparisons. In addition, to address the problem of rewording, Aplag replaces each word's root by the most frequent synonym extracted from Arabic WordNet [31].

Jadalla and Elnagar [32] introduced a plagiarism detection system for Arabic text-based documents named Iqtebas. It uses a fingerprint search engine to compute the distance between each sentence in the suspected text and the closest sentence in the source documents. Iqtebas seems to perform well the copy-and-paste (C&P) plagiarism, but it handles neither word shuffling nor rewording.

Recently, Hussein [33] proposed a new plagiarism detection system for Arabic documents based on modeling the relation between texts and their n-gram unique sentences. The system involves several steps, including Part-of-Speech (POS) tagging, text indexing, stop-words removal, synonyms substitution and heuristic pairwise phrase matching algorithm to build documents Term Frequency-Inverse Document Frequency (TF-IDF) model [45]. The Latent Semantic Analysis (LSA) [46] and Singular Value Decomposition (SVD) [47] are then used to analyse the hidden associations between text documents.

The Arabic Plagiarism Detection Shared Task 2015 (AraPlagDet)[2] [34] is the first and only shared task that addresses the evaluation of plagiarism detection methods for Arabic texts. It has two sub-tasks: extrinsic and intrinsic plagiarism detection. A major advantage of the AraPlagDet evaluation campaign is enabling the evaluation of different systems on the same dataset. In AraPlagDet 2015 three systems are participated in the extrinsic plagiarism detection subtask: Magooda [35], Alzahrani

[2] http://misc-umc.org/AraPlagDet/

[36] and Palkovskii[3]. Two participants (Magooda and Alzahrani) among the three submitted working notes describing their systems.

Magooda et al. [35] proposed an extrinsic plagiarism detection system named RDI_RED. In this system, Lucene search engine [44] is used to select a list of candidate source documents. The candidate documents are aligned to detect plagiarised segments (aligned parts). Finally, a set of rules is applied by a filtering module in order to filter the aligned parts. RDI_RED system can be easily deployed on-line. Though, it does not address synonyms substitution and paraphrasing.

Alzahrani's [36] system goes through four main steps: (1) Pre-processing which includes tokenization and stop-word removal, (2) retrieve a list of candidate source documents for each suspicious document using n-gram fingerprinting and Jaccard coefficient, (3) An in-depth comparison between the suspicious documents and the associated source candidate documents using k-overlapping approach [30], (4) Post-processing where consecutive n-grams are joined to form united plagiarised segments. **Table 1** summarizes the Arabic plagiarism detection systems described above according to the technique used, the comparison level and their efficiency in detecting different plagiarism types.

Table1 Details of the Arabic plagiarism detection systems.

		Systems							
		FS-APD [30]	SFS-APD [32]	Aplag [6]	Iqtebas [34]	Hussein [35]	RDI-RED [37]	Alzahrani [38]	**2L-APD**
Technique	Fingerprinting			×	×			×	×
	Fuzzy-set	×	×						
	SVD					×			
	LSA					×			
	Search Engine						×		
	Linguistic Resources		×	×		×			
	Word Embedding								×
Comparison Level	Sentence-Level	×	×	×	×	×	×	×	×
	Paragraph - Level			×			×		
Plagiarism Type	C&P	×	×	×	×	×	×	×	×
	Reordering	×	×	×	×	×	×	×	×
	Synonyms substitution		×	×		×			×
	Paraphrasing								×

[3] http://plagiarism-detector.com/

3. Background

3.1. Fingerprinting

Fingerprinting is widely applied extrinsic plagiarism [29]. The purpose is to reduce the size of the compared texts and speed up the comparison without missing a significant match. A document fingerprint is a list of integers resulting from hashing substrings of the document. The comparison is then performed on the fingerprint rather than the whole text [11]. The process of creating a fingerprint involves three steps:

- **Chunking:** the document is segmented into substrings (called chunks or minutiae). A chunk might be a sequence of letters, words or even sentences.
- **Hashing:** a hash function is applied to the chunks to generate list of integers.
- **Selection:** The final fingerprint is a subsequence of the list of hashes.

There are four factors which must be carefully balanced when constructing a fingerprint: the fingerprint granularity, the hash function, the selection strategy and the fingerprint resolution [12]:

- *Fingerprint Granularity*

The size of chunk determines the fingerprint granularity, and they have a significant impact on the accuracy of fingerprint [11]. Large chunks fingerprint (coarse granularity), is fast to compute but highly sensitive to changes, whereas small chunks fingerprint (fine granularity) is less sensitive to such changes, yet they require significant computational effort and allows a higher rate of false positive.

- *Hash Function*

A hash function maps the chunks to integers. It is especially important to choose the hash function in such a way as to minimize the collisions due to mapping different chunks to the same hash [11].

- *Selection Strategy*

While hashing all chunks is likely to be the best choice for strict matching, keeping only a subsequence of the checks has shown to be more efficient and less sensitive to insignificant changes [11]. A number of chunk selection approaches have been used so far such as "i^{th} hash" [15], "0 mod k" [13], first-k [12], first-k-sliding strategy [12] and winnowing [14].

- *Fingerprint Resolution*

The number of the selected hashes to represent a document defines the fingerprint resolution. The processing and the storage requirements increase proportionally with the fingerprint resolution [14].

3.2. Word Embedding

Recently, word embedding representation has received a lot of attention in the NLP community and has become a core building to many NLP applications, such as information retrieval, plagiarism detection, machine translation, text classification and text summarization. Word embedding represents words as vectors in a continuous high-dimensional space. Indeed, these representations allow to capture the syntactic and semantic properties of the language [20]. Most word embedding techniques are relying on the neural network to train the word vectors from a large collection of text documents. In the literature, several techniques are proposed to build a word embedding model, among the most famous are: Collobert and Weston model [21], Hierarchical Log-Bilinear Model (HLBL) [22], Turian et al. model [23], Recurrent Neural Network (RNN) model [24], Continuous Bag-of-Words model (CBOW) [20], Skip-gram model (SKIP-G) [25] and Global Vectors model (GloVe) [26].

In a comparative study conducted by Mikolov et al. [20] all the methods [21], [22], [23], [24], and [25] have been evaluated and compared, and they show that CBOW and SKIP-G are significantly faster to train with better accuracy. For this reason, we have used the CBOW word representations for Arabic model proposed by Zahran et al. [19]. To train this model, they have used a large collection from different sources containing more than 5.8 billion words[4]. In this model, each word *w* is represented by a vector *v* of *300-dimension*. The similarity between two words w_i and w_j is obtained by comparing their vector representations v_i and v_j respectively [20]. This similarity can be evaluated using the Cosine similarity, Manhattan distance, Euclidean distance or any other similarity measure functions. For example: let "الجامعة" (*university*), "المساء" (*evening*) and "الكلية" (*faculty*) be three words. The similarity between them is measured by computing the cosine similarity between their vectors as follows:

$$\begin{cases} Sim(\text{المساء}, \text{الجامعة}) = Cos\left(v\left(\text{المساء}\right), v\left(\text{الجامعة}\right)\right) = 0.13 \\ Sim\left(\text{الكلية}, \text{الجامعة}\right) = Cos\left(v\left(\text{الجامعة}\right), v\left(\text{الكلية}\right)\right) = 0.72 \end{cases}$$

This means that the words "الكلية" (*faculty*) and "الجامعة" (*university*) are semantically closer than "المساء" (*evening*) and "الجامعة" (*university*).

4. Proposed System

In order to detect different types of plagiarism, our proposed 2L-APD system is based on two modules (levels): Fingerprinting detection module and Word embedding detection module. The fingerprinting module is designed to detect the literal plagiarism (lexical level), such as C&P, reordering of words and adding filler words. However, in the practical plagiarism cases especially in scientific research, several intelligent plagiarism forms are used, including obfuscations, synonym replacement and paraphrasing. These techniques often generate a significant change

[4] https://sites.google.com/site/mohazahran/data

in the structure of the original text, which can affect considerably the document fingerprint. This fact makes the fingerprinting module quite weak against textual modification. To address this issue, we have proposed a word embedding module (semantic level). If the plagiarism is not detected in the fingerprinting module, the suspect document is sent to the word embedding module to detect intelligent plagiarism. **Figure1** illustrates an overview of 2L-APD system.

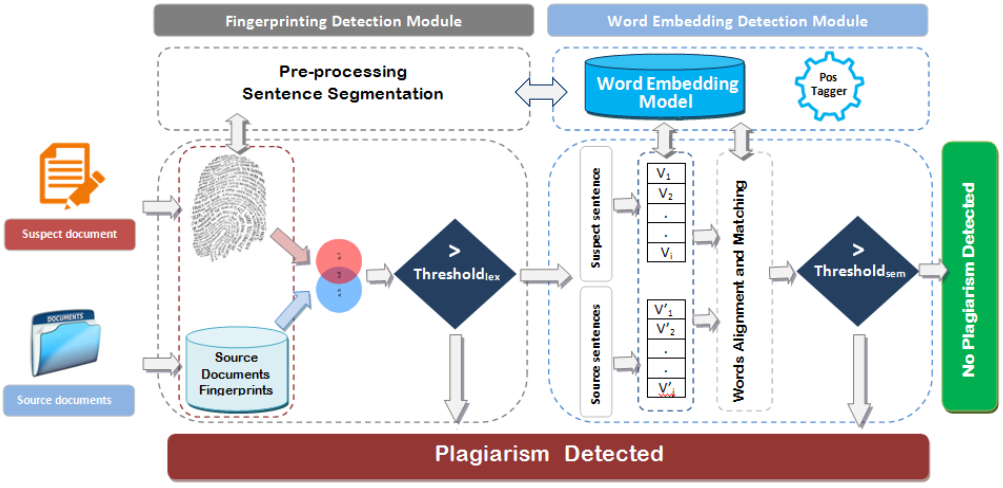

Figure 1: Overview of 2L-APD system

Let $D = \{d_1, d_2, ..., d_i\}$ be a set of potential source of plagiarism documents and let d_{sus} denote a suspicious document. The main task of a plagiarism detector consists in locating the highly similar pairs of passages (p, p') from d_{sus} and d_{src} ($d_{src} \in D$). These passages could have many levels of similarity, such as p' is exactly similar to p, p' is obtained from p by obfuscation techniques or p' and p are semantically similar. In the following, we develop our proposed modules and we provide for each one how the plagiarism detection is performed.

3.1 Segmentation and Pre-processing

In a first step, each document d_{sus} and d_{src} is chunked into sentences. The average length of Arabic sentence is widely higher than other languages, it is around 35 words per sentence [38]. Therefore, we have chosen to use (.), (,), (;), (:), (!) and (?) punctuation marks as a segmentation point, provided that the sentence length should be between 25 and 35 words. In order to normalize the sentences for detection modules, a set of preprocessing steps are applied:

- Tokenization: decompose each sentence into a set of tokens (words).
- Remove diacritics and non-letters.
- Stop-words removal.
- Lemmatization: MADAMIRA tool [37] is used only for the fingerprinting module to reduce words to their lemma, however, in the word embedding module, we use the normal form of words to capture the semantic properties.

3.2 Fingerprinting Detection Module

Detecting plagiarism between a suspicious (d_{sus}) and source document ($d_{scr} \in D$) in the fingerprinting module is carried out as follows:

1. **Fingerprinting**: we construct for each sentence its fingerprint as follows:
   - **Chunking**: each sentence is broken into a set of n-grams (character-based).
   - **Selection**: in this step, we propose a new selection strategy based on our previous work presented in [16]. The key idea of [16] is how to effectively exploit the uneven distribution of the n-grams frequencies in natural language text, to reduce the n-gram inverted index size, where we store only the less frequent n-grams. In fact, we proved that the least frequent n-grams are the most significant. Let us illustrate this fact by considering the problem of searching the word 'dozen' within the Gutenberg corpus [17]. The sequential search suggests starting either with the first letter '*d*', or the last one '*n*'. If we take the first choice, the text is scanned letter by letter until a match with '*d*' is found. If so, the process compares the remaining letters in the word one by one with those in the text until a full match is verified or a mismatch is faced. Since the frequency of the letter '*d*' in our case is 387,163. Starting the search by checking the first letter implies that we will look further nearly 387,163 times at the letter just next to '*d*' to check if it is an '*o*'. However, the letter '*z*' appears no more than 4,735 times in the text. So, if the matching starts from the letter '*z*' almost 99% of the fruitless extra comparisons are avoided. In this way, we propose to select only the n-grams having a frequency smaller than the sampling threshold (T_{smp}).
   - **Hashing**: the Brian Kernighan and Dennis Ritchie (BKDR) [39] hash function is applied to the selected n-grams to generate the sentence fingerprint.
2. **Plagiarism Detection:** measuring the similarity between two documents is carried out by comparing their sentences fingerprints using the Jaccard similarity. Then, the similarity is compared to a fixed threshold (T_{lex}) to judge whether the existence of a shared text and suggest potential plagiarism. If the similarity is lower than T_{lex}, then the suspect sentence is sent to the word embedding module to detect a potential intelligent plagiarism.

3.3 Word Embedding Detection Module

Plagiarism detection at the semantic level is carried out as follows: let $S_{src}= w_1 ,w_2,...,w_i$ and $S_{sus}=w'_1,w'_2,\ldots, w'_j$ be a source and suspect sentences, their word vectors in the Arabic CBOW are $(v_1,v_2,\ldots,v_i)$ and $(v'_1,v'_2,\ldots,v'_j)$ respectively. A simple method to compare two sentences is to sum their word vectors [40]. Then, the similarity between S_{src} and S_{sus} is obtained by calculating the cosine similarity between V_{src} and V_{sus}, where: $V_{src} = \sum_{k=1}^{j} v_k$, $V_{sus} = \sum_{k=1}^{i} v'_k$. For example, S_{src} S_{src}= "ذهب يوسف إلى الكلية" (*Joseph went to college*) and S_{sus}= "يمضى يوسف مسرعا للجامعة" (*Joseph goes quickly to university*).

The similarity between S_{src} and S_{sus} is obtained by calculating the Cosine similarity $Cos(V_{src}, V_{sus})$, where:

$$\begin{cases} V_{src} = v(\text{ذهب}) + v(\text{يوسف}) + v(\text{إلى}) + v(\text{الكلية}) \\ V_{sus} = v(\text{للجامعة}) + v(\text{مسرعا}) + v(\text{يوسف}) + v(\text{يمضي}) \end{cases}$$

The similarity between S_{src} and S_{sus} is obtained by calculating the cosine similarity between their sentence vectors V_{src} and V_{sus} as follows:

$$Sim(S_{src}, S_{sus}) = Cos(V_{src}, V_{sus}) = 0.71$$

In order to improve the similarity results, we have used the word alignment method presented by Sultan et al. [41], with the difference that we align the words based on their semantic similarity in the word embedding model. We assume also that the words do not have the same importance for the meaning of the sentences. For that, we have used two weighting functions (IDF and POS) proposed by El Moatez and Didier in [40] to weight the aligned words. Then, the similarity between S_{src} and S_{sus} is measured by the formula (1):

$$(1)\quad Sim(S_{src}, S_{sus}) = \frac{1}{2}\left(\frac{\sum_{w_k \in S_{src}} WT(w_k) * BM(w_k, S_{sus})}{\sum_{w_k \in S_{src}} WT(w_k)} + \frac{\sum_{w_k \in S_{sus}} WT(w_k) * BM(w_k, S_{src})}{\sum_{w_k \in S_{sus}} WT(w_k)}\right)$$

where $WT(w_k)$ is a mix of both IDF and POS weight of w_k, and $BM(w_k, S_x)$ is the *Best Match* score between w_k and all words in the sentence S_x. The *BM* function aligns words based on their semantic similarity, *BM* is defined as:

$$(2)\quad BM(w_k, S_k) = Max\{\ Cos(v_k, v_r),\quad v_r \in S_x\quad\}$$

Finally, the similarity $Sim(S_{src}, S_{sus})$ is compared to a second fixed threshold (T_{sem}) to judge whether the existence of a potential plagiarism. Let us continue with the same example above. The similarity between S_{src} and S_{sus} is obtained in four steps:

1. **POS Tagging** in this step the POS tagger of G. Braham et al. [42] is used to estimate the POS of each word w_k in S_k,

$$\begin{cases} Pos\_tag(S_{src}) = verb\ noun\_prop\ noun \\ Pos\_tag(S_{sus}) = noun\_prop\ verb\ adj\ noun \end{cases}$$

2. **Word Alignment** In this step, we align words that have similar meaning in both sentences. For that, we compute the similarity between each word in S_{src} and the semantically closest word in S_{sus} by using the BM function, e.g. $BM(\text{يمضي}, S_{src}) = Max\{Cos(\text{يمضي}, v_k),\quad w_k \in S_{sus}\} = Cos(v(\text{يمضي}), v(\text{ذهب}))$.

3. **IDF & POS Weighting** In order to weight the descriptive aligned words, we retrieve for each word w_k in the S_x its IDF weight $idf(w_k)$, we also use the POS weights proposed in [40]. The weight of each word w_k is obtained as follows: $WT(w_k) = idf(w_k) * Pos\_weight(w_k)$, where $Pos\_weight(w_k)$ is the function which return the weight of POS tagging of w_k.

4. **Calculate the similarity** the similarity between S_{src} and S_{sus} is obtained by using the formula (1), which gives us: $Sim(S_{src}, S_{sus}) = Cos(V_{src}, V_{sus}) = 0.85$

5. Experiments and Results

5.1. Data Set

In order to evaluate our system and monitor its performance against other systems on the same dataset, we have used the "*External Arabic Plagiarism Corpus*" (ExAra-2015)[5]. This corpus is released as part of the AraPlagDete Shared Task 2015 [34]. The ExAra-2015 corpus contains two sets of documents: (1) the source documents, from which passages of text are extracted; and (2) the suspicious documents, in which the plagiarised passages are inserted directly or after undergoing obfuscation processing. The suspicious documents contain two kinds of plagiarism cases: artificial (created automatically) and simulated (created manually). The first one, use two types of obfuscation, *phrase shuffling* and *word shuffling*. The manually created plagiarism simulates a real plagiarism cases by using a manual synonym substitution, diacritics insertion and paraphrasing. More details about the ExAra-2015 and the obfuscation used are given in **Table 2** and **3** respectively.

5.2. Performance Measures

The performance of our 2L-APD system is quantified by the character-based macro *recall* and *precision*, supplemented by two other measures proposed in [43] called *granularity* and *plagdet*. These measures are computed using the two sets: plagiarism cases annotated in the corpus ***S*** (*actual cases*) and the cases detected by our system ***R*** (*detected cases*). Let d_p be a document that contains plagiarism. A *plagiarism case* in d_p is a 4-tuple $s \in S$, where $s = < s_p, d_p, s_{sr}, d_{sr} >$, s_p is a plagiarized passage in d_p, and s_{sr} is its original passage in some source document d_{sr}. Let $r \in R$ denote a *plagiarism detection* for the document d_p, where $r = < r_p, d_p, r_{sr}, d'_{sr} >$, r_p is a potential plagiarized passage in d_p, and r_{sr} its source d'_{sr}. We say that, ***s*** is ***detected*** by ***r*** *iff* $d_{sr} = d'_{sr}$, $r_p \cap s_p \neq \emptyset$ and $r_{sr} \cap d_{sr} = \emptyset$.

- **Recall and Precision**: recall and precision is the fraction of the true positive part in each actual and detected case respectively. Their formulas are given in the equations (2) and (3).

(2) $Recall(S, R) = \frac{1}{|S|} \sum_{s \in S} \frac{\bigcup_{r \in R}(s \sqcap r)}{|s|}$; (3) $Precision(S, R) = \frac{1}{|R|} \sum_{r \in R} \frac{\bigcup_{s \in S}(s \sqcap r)}{|r|}$

$$where\ (s \sqcap r) = \begin{cases} s \cap r & \text{if } r \text{ detects } s \\ \emptyset & \text{otherwise} \end{cases}$$

Neither recall nor precision account for the fact that plagiarism systems may report multiple or overlapping detections for the same plagiarism case. To address this issue, also a granularity detector is used [29].

- **Granularity:** quantifies whether the contiguity between plagiarized text passages is properly recognized [43]. The granularity is depicted in the formula (4).

(4) $$Granularity(S, R) = \frac{1}{|S_R|} \sum_{s \in S_R} |R_S|$$

[5] http://misc-umc.org/AraPlagDet/#datasets

where $S_R \subseteq S$ is the set of the actual cases that have been detected, and $R_S \subseteq R$ are the detections of a given s:

$$S_R = \{s \mid s \in S \wedge \exists r \in R : r \text{ detects } s\}, \quad R_S = \{r \mid r \in R \wedge r \text{ detects } s\}$$

- **Plagdet:** precision, recall, and granularity not allow for an absolute ranking among different system [43], *plagdet* that combines these measures in one measure as expressed in the formula (5).

(5) $$Plagdet(S, R) = \frac{F_1}{(1+Granularity(S, R))}$$

where F_1 is the equally-weighted harmonic mean of recall and precision.

Table 2: Details of ExAra-2015 corpus [34]

Generic Information	Documents number	1171
	Cases number	1727
	Source documents	48.68%
	Suspicious documents	51.32%
Plagiarism per document	Without plagiarism	28.12%
	With plagiarism	71.88%
	Hardly (1%-20%)	36.94%
	Medium (20%-50%)	32.95%
	Much (50%-80%)	2.00%
Length of plagiarism case	Very short (300 chars)	21.25%
	Short (300-1k chars)	42.50%
	Medium (1k-3k chars)	28.26%
	Long (3k-30k chars)	7.99%
Plagiarism type and obfuscation	Artificial	88.94%
	Without obfuscation	40.30%
	Phrase shuffling	10.42%
	Word shuffling	38.22%
	Simulated	11.06%
	Manual synonym substitution.	9.79%
	Manual paraphrasing	1.27%

Table 3: Types of plagiarism and obfuscation used in ExAra-2015 corpus

Type	Description
Manual Synonym Substitution	Replaced some words with their synonyms by using the Microsoft Word synonym checker, Almaany dictionary, Arabic WordNet Browser, and the synonyms provided by Google translate.
Added and/or removed diacritics	Diacritics in Arabic are optional and their exclusion or inclusions are orthographically acceptable. For example: القَضيّةُ الفِلَسطينيةُ ≡ القَضيّةُ الفِلَسْطينيّةُ ≡ القَضِيَّةُ الفِلَسْطِينِيّةُ ≡ القضية الفلسطينية ≡ …
Automatically obfuscation	Phrase shuffling and word shuffling strategy are used to create automatically obfuscation cases, e.g. القضية الفلسطينية مصطلح يشار به للخلاف السياسي والتاريخي ← يشار مصطلح الفلسطينية القضية به للخلاف التاريخي والسياسي
Manual Paraphrasing	The passages to be obfuscated are manually selected from the source documents then paraphrased manually, e.g. القضية الفلسطينية مصطلح يشار به للخلاف السياسي و التاريخي و الأزمة الانسانية في فلسطين بدءا من عام ١٨٤٠ ← بدء الخلاف السياسي في فلسطين منذ أواخر القرن التاسع عشر مما أدى إلى أزمة إنسانية أصبحت تعرف بالقضية الفلسطينية

5.2. Thresholds

Before presenting the results, we should mention that the sampling (T_{smp}), lexical (T_{lex}) and semantic (T_{sem}) thresholds are empirically fixed using the training data of the AraPlagDet 2015 (*Tr-ExAra-2015 corpus*) [34]. In Tr-ExAra-2015 each suspicious document is associated with an XML document that locates the exact position of the plagiarised passages. Additionally, the suspicious documents are classified into four sets according to the type of plagiarism used which include: without plagiarism, C&P plagiarism, artificial plagiarism (phrase shuffling and word shuffling) and simulated plagiarism (synonym substitution, added diacritics and paraphrasing). In fact, we have used the C&P and artificial plagiarism cases to determine the lexical threshold value T_{lex} and the simulated plagiarism cases for the semantic threshold T_{lex}. Thus, T_{lex} is set to 15%, which means that two fingerprints describing two different sentences have an intersection less than 15%, and T_{sem} is set to 60% to indicate a potential intelligent plagiarism. Regarding the sampling threshold T_{smp}, it is adjusted according to n-gram size used. As we have chosen to use 3-gram as a unit of chunk, T_{smp} is set to 0.008%, 0.01%, and 0.05% respectively for selecting 10%, 20% and 50% of all 3-grams.

5.3. Results

Several variants of 2L-APD were tested to measure the impact of the fingerprint resolution and the word embedding level on the detection accuracy. The values of the *precision, recall , granularity* and *plagdet* for different fingerprint resolution: Fine (F), Medium (M) and Coarse (C) (10%, 20% and 50% off all 3-grams are selected receptively), with and without the Word Embedding (WE) detection module are shown in Table 4. The results obtained can be summarized as follows: when the fingerprint resolution is Fine, the precision is reasonable where 73% of detected cases were correct, but the recall is very low and equal to about 43%. When applying the Medium resolution the precision increases slightly to 79%, however, the recall is greatly enhanced to 62%. This is due to increased number of n-grams selected in the fingerprint (i.e. more information is encoded and used as indicative of reused text segments). For the coarse resolution, the rate of increase is not significant compared to the Medium. This means that the medium resolution is able to encode sufficient information about the documents to ensure the detection. Interestingly, employing the word embedding model significantly enhances the recall (with a mean of +24.3%). This is due to the inability to detect the intelligent plagiarism in the fingerprint model.

Table 4: Performance of the 2L-APD on the ExAra-2015 corpus

Method	Precision	Recall	Granularity	Plagdet
FP(F)	0.7315	0.4347	**1.055**	0.5255
FP(M)	0.7713	0.6251	1.058	0.6631
FP(C)	0.7856	0.6383	1.059	0.6882
FP(F)+WE	0.7521	0.6623	1.057	0.6769
FP(M)+WE	**0.8593**	0.8781	1.064	**0.8308**
FP(C)+WE	0.8413	**0.8867**	1.068	0.8236

5.4. Comparison

We have compared our best method ***FP(M)+WE*** to the ones obtained by Magooda (3 methods) [35], Alzahrani [36], Palkovskii (3 methods) and the baseline [34]. Table 5 shows the overall performances of the plagiarism detectors methods that were tested on the ExAra-2015 corpus. As expected, in terms of the recall, plagdet and granularity our method outperforms the baseline. The overall best performing method is the ***FP(M)+WE*** with a gain of +2.89% on plagdet. In term of recall, ***FP(M)+WE*** leads to an overall recall score of 87.81% against 83.10% for Magooda(2). The low recall of other methods due to their inability to detect some obfuscation plagiarism cases like manual paraphrasing.

Table 5: Comparison Results

Method	Precision	Recall	Granularity	Plagdet
FP(M)+WE	**0.859**	**0.878**	1.064	**0.831**
Magooda(2)	0.852	0.831	1.069	0.802
Magooda(3)	0.854	0.759	1.058	0.772
Magooda(1)	0.805	0.786	**1.052**	0.767
Palkovskii(1)	0.997	0.542	1.062	0.627
Baseline	**0.990**	0.535	1.209	0.608
Alzahrani	0.831	0.530	1.186	0.574
Palkovskii(3)	0.658	0.589	1.161	0.560
Palkovskii(2)	0.564	0.589	1.163	0.518

6. Conclusion and Future Work

In this paper, we presented a Plagiarism Detection system acting at two layers: fingerprinting and word embedding. At the first layer, the system computes the fingerprints of all sentences in the source and suspect documents. The comparison is then performed between fingerprints rather than original texts. Our contribution to such classical approach of plagiarism detection is the introduction of a novel selection strategy in which the statistical characteristics of the natural text are used to select only the less frequent n-grams as a fingerprint.

To push the capabilities of the system further to handle more advanced plagiarism cases such obfuscations, synonym substitution and paraphrasing. The second layer uses the semantic properties of words characterised in the word embedding combined with word alignment, IDF and POS weighting to support the identification of the words that are the most descriptive in each textual units.

The performance of the system is confirmed in terms of recall which reached 88% and precision with 85%. Our system outperformed all systems participating in the Arabic Plagiarism Detection Shared Task 2015 with a plagiarism detection score of 83%. The tests show clearly the ability of the system to handle various types of plagiarism including literal plagiarism, reordering, rewording, synonym substitution and paraphrasing.

As our method consists in cutting up the document into sentences, an improvement would be to use a sentence2vec model instead of a word2vec model. We would also like to further investigate the plagiarism detection task with more sophisticated methods, such as Recurrent Neural Network (RNN) and Convolutional Neural Networks (CNN) trained on a pre-trained word/sentence embedding vectors.

While the investigation has been conducted on one application namely the plagiarism detection, intuition suggests that an efficient assessment of shared information is applicable to other applications such as authorship classification, semantic similarity and sentiment analysis. An obvious elaboration would be to investigate the performance of the system once adapted to other languages.

References

1. Teddi Fishman. “we know it when we see it” is not good enough: toward a standard definition of plagiarism that transcends theft, fraud, and copyright. 2009.
2. Bela Gipp. Citation-based plagiarism detection. In Citation-based Plagiarism Detection, pages 57–88, Springer, 2014.
3. Pascal Guibert and Christophe Michaut. Le plagiat étudiant. Education et sociétés (2):149–163, 2011.
4. Donald L McCabe. Cheating among college and university students: A north American perspective. International Journal for Educational Integrity, 1(1), 2005.
5. AS Bin-Habtoor and MA Zaher. A survey on plagiarism detection systems. International Journal of Computer Theory and Engineering, 4(2):185, 2012.
6. Mohamed El Bachir Menai. Detection of plagiarism in Arabic documents. International journal of information technology and computer science (IJITCS), 4(10):80, 2012.
7. Farghaly, A., & Shaalan, K. (2009). Arabic natural language processing: Challenges and solutions. ACM Transactions on Asian Language Information Processing (TALIP), 8(4), 14.
8. Chao Liu, Chen Chen, Jiawei Han, and Philip S Yu. Gplag: detection of software plagiarism by program dependence graph analysis. In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 872–881. ACM, 2006.
9. Salha M Alzahrani, Naomie Salim, and Ajith Abraham. Understanding plagiarism linguistic patterns, textual features, and detection methods. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 42(2):133–149, 2012.
10. 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 CLEF Conference on Multilingual and Multimodal Information Access Evaluation, pages 301–331. CELCT, 2013.
11. Benno Stein and Sven Meyer Zu Eissen. Near similarity search and plagiarism analysis. In From data and information analysis to knowledge engineering, pages 430–437. Springer, 2006.
12. Timothy C Hoad and Justin Zobel. Methods for identifying versioned and plagiarized documents. Journal of the Association for Information Science and Technology, 54(3):203–215, 2003.
13. Manber Udi, Finding similar files in a large file system, In Proceedings of the USENIX Winter 1994 Technical Conference, 1994.
14. Saul Schleimer, Daniel S Wilkerson, and Alex Aiken. Winnowing: local algorithms for document fingerprinting. In Proceedings of the 2003 ACM SIGMOD international conference on Management of data, pages 76–85. ACM, 2003.
15. Richard M Karp and Michael O Rabin. Efficient randomized pattern-matching algorithms. IBM Journal of Research and Development, 31(2):249–260, 1987.
16. El Moatez Billah Nagoudi, Ahmed Khorsi, and Hadda Cherroun. Efficient inverted index with n-gram sampling for string matching in arabic documents. In The 13th IEEE/ACS International Conference on Computer Systems and Applications, Agadir, Morocco, 2016, pp. 1-7.

17. Marie Lebert. Project Gutenberg (1971-2008). Project Gutenberg, 2008.
18. Yasushi Ogawa, Tetsuya Morita, and Kiyohiko Kobayashi. A fuzzy document retrieval system using the keyword connection matrix and a learning method. Fuzzy sets and systems, 39(2):163–179, 1991.
19. Mohamed A Zahran, Ahmed Magooda, Ashraf Y Mahgoub, Hazem Raafat, Mohsen Rashwan, and Amir Atyia. Word representations in vector space and their applications for arabic. In International Conference on Intelligent Text Processing and Computational Linguistics, pages 430–443. Springer, 2015.
20. Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In In: ICLR: Proceeding of the International Conference on Learning Representations Workshop Track, pages 1301–3781, 2013.
21. Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In Proceedings of the 25th international conference on Machine learning, pages 160–167. ACM, 2008.
22. Mnih, A., & Hinton, G. E. (2009). A scalable hierarchical distributed language model. In Advances in neural information processing systems (pp. 1081-1088).
23. Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: a simple and general method for semi-supervised learning. In Proceedings of the 48th annual meeting of the association for computational linguistics, pages 384–394. Association for Computational Linguistics, 2010.
24. Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In Hlt-naacl, volume 13, pages 746–751, 2013.
25. Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems, pages 3111–3119, 2013.
26. Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In EMNLP, volume 14, pages 1532–1543, 2014.
27. Hermann A Maurer, Frank Kappe, and Bilal Zaka. Plagiarism a survey. J. UCS, 12(8):1050–1084, 2006.
28. Salha Mohammed Alzahrani and Naomie Salim. Plagiarism detection in arabic scripts using fuzzy information retrieval. In Student Conf., Johor Bahru, Malaysia, pages 281–285, 2008.
29. 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 CLEF Conference on Multilingual and Multimodal Information Access Evaluation, pages 301–331. CELCT, 2013.
30. Salha Alzahrani and Naomie Salim Fuzzy semantic-based string similarity for extrinsic plagiarism detection: Lab report for PAN at CLEF'10," presented at the 4th Int. Workshop PAN-10, Padua, Italy, 2010.
31. William Black, Sabri Elkateb, Horacio Rodriguez, Musa Alkhalifa, Piek Vossen, Adam Pease, and Christiane Fellbaum. Introducing the arabic wordnet project. In Proceedings of the third international Word- Net conference, pages 295–300, 2006.
32. Ameera Jadalla and Ashraf Elnagar. A plagiarism detection system for arabic text-based documents. In Pacific-Asia Workshop on Intelligence and Security Informatics, pages 145–153. Springer, 2012.
33. Ashraf S Hussein. A plagiarism detection system for arabic documents. In Intelligent Systems' 2014, Springer International Publishing, 2015. p. 541-552.
34. Imene Bensalem, Imene Boukhalfa, Paolo Rosso, Lahsen Abouenour, Kareem Darwish, and Salim Chikhi. Overview of the araplagdet pan@ fire2015 shared task on arabic plagiarism detection. In FIRE Workshops, pages 111–122, 2015.
35. Ahmed Magooda, Ashraf Y Mahgoub, Mohsen Rashwan, Magda B Fayek, and Hazem M Raafat. Rdi system for extrinsic plagiarism detection (rdi red), working notes for panaraplagdet at fire 2015. In FIRE Workshops, pages 126–128, 2015.
36. Salha Alzahrani. Arabic plagiarism detection using word correlation in n-grams with k-overlapping approach, working notes for panaraplagdet at fire 2015 workshops, p 123–125, 2015.

37. Arfath Pasha, Mohamed Al-Badrashiny, Mona T Diab, Ahmed El Kholy, Ramy Eskander, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan Roth. Madamira: A fast, comprehensive tool for morphological analysis and disambiguation of arabic. In LREC, v 14, pages 1094–1101, 2014.
38. Ryan McDonald, Kevin Lerman, and Fernando Pereira. Multilingual dependency analysis with a two-stage discriminative parser. In Proceedings of the Tenth Conference on Computational Natural Language Learning, pages 216–220. Association for Computational Linguistics, 2006.
39. Dennis M Ritchie, Brian W Kernighan, and Michael E Lesk. The C programming language. Prentice Hall Englewood Cliffs, 1988.
40. El Moatez Billah Nagoudi and Didier Schwab. Semantic Similarity of Arabic Sentences with Word Embeddings. In Proceedings of the Third Arabic Natural Language Processing Workshop (WANLP), pages 18–24. Association for Computational Linguistics, 2017.
41. Md Arafat Sultan, Steven Bethard, and Tamara Sumner. Dls@ cu: Sentence similarity from word alignment and semantic vector composition. In Proceedings of the 9th International Workshop on Semantic Evaluation, pages 148–153, 2015.
42. Souhir Gahbiche-Braham, Hélene Bonneau-Maynard, Thomas Lavergne, and Franc¸ois Yvon. Joint segmentation and pos tagging for arabic using a crf-based classifier. In LREC, pages 2107–2113, 2012.
43. Martin Potthast, Benno Stein, Alberto Barrón-Cedeño, and Paolo Rosso. An evaluation framework for plagiarism detection. In Proceedings of the 23rd international conference on computational linguistics: pages 997–1005.Association for Computational Linguistics, 2010.
44. McCandless, M., Hatcher, E., & Gospodnetic, O. (2010). Lucene in Action: Covers Apache Lucene 3.0. Manning Publications Co.
45. Salton, G. (1989). Automatic text processing: The transformation, analysis, and retrieval of. Reading: Addison-Wesley.
46. Sabrina Simmons and Zachary Estes. Using latent semantic analysis to estimate similarity. In Proceedings of the Cognitive Science Society, pages 2169–2173, 2006
47. Zdenek Ceska. Plagiarism detection based on singular value decomposition. In Advances in natural language processing, pages 108–119. Springer, 2008.