Journal of Organizational and End User Computing
Volume 32 • Issue 2 • April-June 2020

Unsupervised Model for Detecting Plagiarism in Internet-based Handwritten Arabic Documents

Mahmoud Zaher, Faculty of Computers and Information, Mansoura University, Mansoura, Egypt
Abdulaziz Shehab, Faculty of Computers and Information, Mansoura University, Mansoura, Egypt
Mohamed Elhoseny, Faculty of Computers and Information, Mansoura University, Mansoura, Egypt
Farahat Farag Farahat, Sadat Academy, Cairo, Egypt

ABSTRACT

Due to the rapid increase of internet-based data, there is urgent need for a robust intelligent documents security mechanism. Although there are many attempts to build a plagiarism detection system in natural language documents, the unlimited variation and different writing styles of each character in Arabic documents make building such systems challenging. Based on its position in a word, the same Arabic letter can be written three different ways, which makes the handwritten character recognition a cumbersome process. This article proposes an intelligent unsupervised model to detect plagiarism in these documents called ASTAP. First, a handwritten Arabic character recognition system is proposed using the Grey Wolf Optimization (GWO) algorithm. Then, a modified Abstract Syntax Tree (AST) is used to match the contents of the Arabic documents to detect any similarity. Compared to the state-of-the-art methods, ASTAP improves the effectiveness of the plagiarism detection in terms of the matched similarity ratio, the precision ratio, and the processing time.

KEYWORDS

Abstract Syntax Tree, Gray Wolf Optimization, Handwritten Character Recognition, Hash value, Internet Data Security, Plagiarism Detection, Similarity Index, Unsupervised Documents Analysis

AN INTRODUCTION

The ever-increasing smart information processing services and applications offered by the Internet have explosively widened the span of the global inter-network. The recent advancements in designing low-cost small scaled devices have harbingered a great surge in the number of Internet-enabled devices which generate a big amount of data. Accordingly, internet data management for discovering plagiarized documents plays a vital role in many applications such as file management, copyright saving, and electronic theft prevention (Lam, et al., 2016; Abdi et al., 2015). Plagiarism not only depends on the content ratio that is copied but dramatically relates to using the work of others, i.e., ideas; without proper citation (Kahloula & Berri, 2016; Abdelrahman & Khalid, 2014).

In Internet-based document processing applications (Chen & Zhao,2017), the Arabic language is considered one of the most complicated languages, especially if the document contains handwritten words. The features of Arabic alphabets have various shapes of the written form based on their position and can be extended by making a dash between the **two** letters. For Arabic in electronic or printed media, no pronouncement makes misunderstanding for some words in an inevitable situation.

DOI: 10.4018/JOEUC.2020040103
This article, originally published under IGI Global's copyright on April 1, 2020 will proceed with publication as an Open Access article starting on January 21, 2021 in the gold Open Access journal, Journal of Organizational and End User Computing (converted to gold Open Access January 1, 2021), and will be distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0/) which permits unrestricted use, distribution, and production in any medium, provided the author of the original work and original publication source are properly credited.

These challenges make the plagiarism detection in Arabic documents an arduous task. Dependently, many machine learning and artificial intelligence based methods have been developed (Hussein, 2016; Wise, 2012). For example, an online Arabic plagiarism detection tool called APD (Alzahrani & Salim, 2015) is proposed to detect the plagiarism on the Arabic web pages. However, this tool does not handle the synonyms alternations or the rewording problem. To avoid that, another system called Plaggie (Ahtiainen et al., 2011) is proposed. Besides its disability to handle the handwritten documents, Plaggie needs a long processing time to manage a computerized Arabic document.

Due to the Hugging of information, and correlation networks, the discovery of electronic thefts is a difficult task, and the discovery of the thefts started in the Arabic language and the most difficult task no doubt. And in light of the growing e-learning systems in the Arab countries, this requires special techniques to detect thefts electronic written in Arabic. And although it could use some search engines like Google, it is very difficult to copy and paste the sentences into the search engines to find these thefts. For this reason, it must develop a good tool for the discovery of electronic thefts written the Arabic language to protect e-learning systems, and to facilitate and accelerate the learning process, where it can automatically detect electronic thefts automatically by this tool.

This paper shows, ASTAP, a system that works on the Internet to enable specialists to detect thefts of electronic texts in Arabic so it can be integrated with e-learning systems to ensure the safety of students and research papers and scientific theses of electronic thefts.

The paper also describes the major components of this system, including stage outfitted, and in the end, we will establish an experimental system on a set of documents and Arabic texts and compared the results obtained with some of the existing systems, particularly TurnItIn.

Accordingly, a new plagiarism_detection system has been proposed in this paper which can handle the internet based handwritten Arabic documents called ASTAP (Abstract Syntax Tree Arabic Plagiarism). ASTAP consists of two main phases. The first one aims to provide an Optical Character Recognition (OCR) tool for internet-based handwritten Arabic Documents. The proposed OCR has two primary functions, feature extraction and feature selection. The feature extraction process aims to remove redundancy from handwritten Arabic characters. While in the selected feature the most relevant are only reserved for improving the accuracy classification. The proposed OCR is implemented using a well-known optimizer called GWO (Seyedali, et al., 2014) that is used to optimize a character features selection. The second phase of ASTAP aims to detect the similarity of Arabic documents using a modified AST (Aiken, 2015; El Bachir & Bagais, 2014). For each node of the AST, the algorithm determines the hash value of AST and compares it with the other nodes in the form of node by node. Also, the algorithm compares sub_trees based on the tree_structure with some reduction in the execution. We have improved the way of syntax tree similarity and proposed a plagiarism detection algorithm that rearranges the nodes of AST to the longitudinal framework. The modified AST consists of five components: AST construction, hash value computation, node classification, hash comparison, and degree of similarity evaluation.

There are two main contributions of that paper. First, an intelligent unsupervised model for internet-based handwritten Arabic character recognition system is proposed using GWO algorithm. Second, a modified AST is proposed for matching the contents of the Arabic documents to detect any similarity. The proposed system improves the similarity accuracy for the plagiarized document by replacing the word synonyms and minimizing time consumption by enhancing the performance of AST algorithm.

The rest of this paper is organized as follow: Section 2 presents an overview of the different related works. Section 3 describes the working steps of the proposed system ASTAP. Section 4 explains the methodology and the internal ASTAP components and their roles in detecting a document similarity. Section 5 presents a discussion of the results. Finally, Section 6 concludes the paper.

RELATED WORKS

As general, many types of research aim to detect documents similarities. Despite the enormous efforts to discover the similarity of Arabic documents, most of the previous work focused the electronic form of these documents. This paper is the first attempt towards detecting plagiarism in internet-based handwritten Arabic documents.

A content-based system (Chen & Zhao, 2017) for analysis and visualization the Arabic documents matching was proposed using Latent Semantic Analysis (LSA) and TF-IDF models. However, this is a simple system that aims to handle light content documents with a limited count of words. Besides, it did not provide a way to handle the synonyms alternations or the rewording problem.

In (Hussein, 2016) a proposed algorithm is used to evaluate the performance of the document visualizations methodologies to detect the relationships inside and between the graphic components of those documents. Recently the essential concepts of design document visualization, and challenges, as well as hopeful sides of future development, have been checked out.

(Subba, 2014) proposed an anti-plagiarism, automated, and flexible grading system for assignments Web-CAT. It works as an e-learning system to test software and helps the students for automatically assess their assignments. Also, the tests offered a number of possible futures adds it does not clear if it is an open source or not.

Plaggie (Ahtiainen, et al., 2011) developed an open-source plagiarism detection system for detecting matches between two source code files. However, if the number of sending files is large, Plaggie takes more time and effort to explain the result.

(Aiken, 2015) developed a web-service Moss. It can measure matches between documents and uses an algorithm called winnowing fingerprinting. The fingerprinting splits a document into hashed sub_strings named k_grams. Next, these fingerprints are used to match couples of programs. As a final point, the results come out as an HTML output on its private server for two weeks and give the URL to the customer.

(El Bachir & Bagais, 2014) APlag is another system for detecting plagiarism in Arabic documents. APlag has three main operations called tokenization, stopwords removing, and roots conversion. Besides its disability to handle the handwritten documents, APlag cannot handle the synonyms alternations problem.

SRL (Osman, et al., 2012) is used to build a plagiarism detection scheme by producing arguments for each sentence semantically. The proposed schema aimed to handle a multi-language document including Arabic. However, an extended analysis is required to evaluate its effectiveness in Arabic documents.

(Grozea & Popescu, 2011) proposed a cross-lingual method for detecting plagiarized documents using an arithmetical model to evaluate the matches between assumed and original documents. Their method uses an English-Spanish dictionary to detect matches in cross-language. In their future work, the authors aim to enlarge their system to include more complicated documents with different languages, such as Arabic documents.

(Mozgovoy & Frederiksson, 2014) developed a detection system with online and offline subsystems. Online subsystem matches detection system which can review text for crumbs that can be found in the web search engines where offline subsystem matches detection system operates on text into a specific collection which can also review the data stored in a local database.

(Chow & Salim, 2013) have developed a structure-based plagiarism detection in programming code called JPlag at Karlsruhe University which is not an open source and consequently cannot be spread by the users. Conversely, JPlag changes codes into some token sequences that basically symbolize the program. Then codes are matched in couples using an algorithm called 'Greedy_String_ Tiling' The outcomes have come as HTML files. JPlag helps the Scheme of some programming languages such as Java, and natural_language manuscript.

(Clough, 2014) have developed an open-source plagiarism detection system Plaggie is used single for Java codes and can be spread. Plaggie outcomes by configuration parameters wanted, Plaggie looks like JPlag, but it is a Java application stand_alone command_line and has to be set up locally. They use tokenization and Greedy String Tiling algorithms for detecting matches between two source code files namely followed by the GST (Greedy_String_Tiling) algorithm. The outcomes recovered as HTML files. However, if the number of sending files is large, Plaggie takes more time and effort to explain the result because it doesn't use grouping to display outcomes instead it displays all matches in a list.

(Borner, et al., 2012) have introduced a system to check the Plagiarism in Arabic documents. It uses tokenization, removes stop-words, and convert the words to their roots in the preprocessing phase, after that the words are switched to their synonyms.

APD (Si, et al., 2012) is proposed as an e-learning tool for Arabic plagiarism detection in web-based documents. APD helps the users of an e-learning system to identify plagiarized online documents. APD allows the teachers to check similarity ratio of the students' assignments that they submitted to the e-learning system after searching online for the most related contents. However, the system does not handle the synonyms alternations or the rewording problem.

Consequently, Turnitin (Alzahrani & Salim, 2008) is a well-known system which is used mostly to evaluate students' works at educational institutions. It uses a cloud-based service for originality checking, online grading, and peer review saves instructors time and provides rich feedback to students. It consists of 3 main tools: PeerMark, GradeMark and OriginalityCheck. However, the system is fragile when handling the synonyms alternations and does not treat the rewording problem. Furthermore, it is too hard to track the similarities of the compared texts.

(Alzahrani & Salim, 2015) proposed a statement-based Arabic plagiarism detection system based on fuzzy-set information retrieval model. Their proposed system is based on computing the similarity between two statements and then comparing it to a threshold value. However, their proposal does not take into account the sentence paraphrasing with different synonyms. Moreover, dealing with the various inflexion forms for the same words were not considered.

ASTAP Components and Working Steps

As mentioned previously, ASTAP provides the ability to handle both handwritten and electronic Arabic documents. As shown at Figure 1, the main operations of ASTAP in these two different cases are mostly the same except only one additional OCR function that is needed to convert the document to its electronic form.

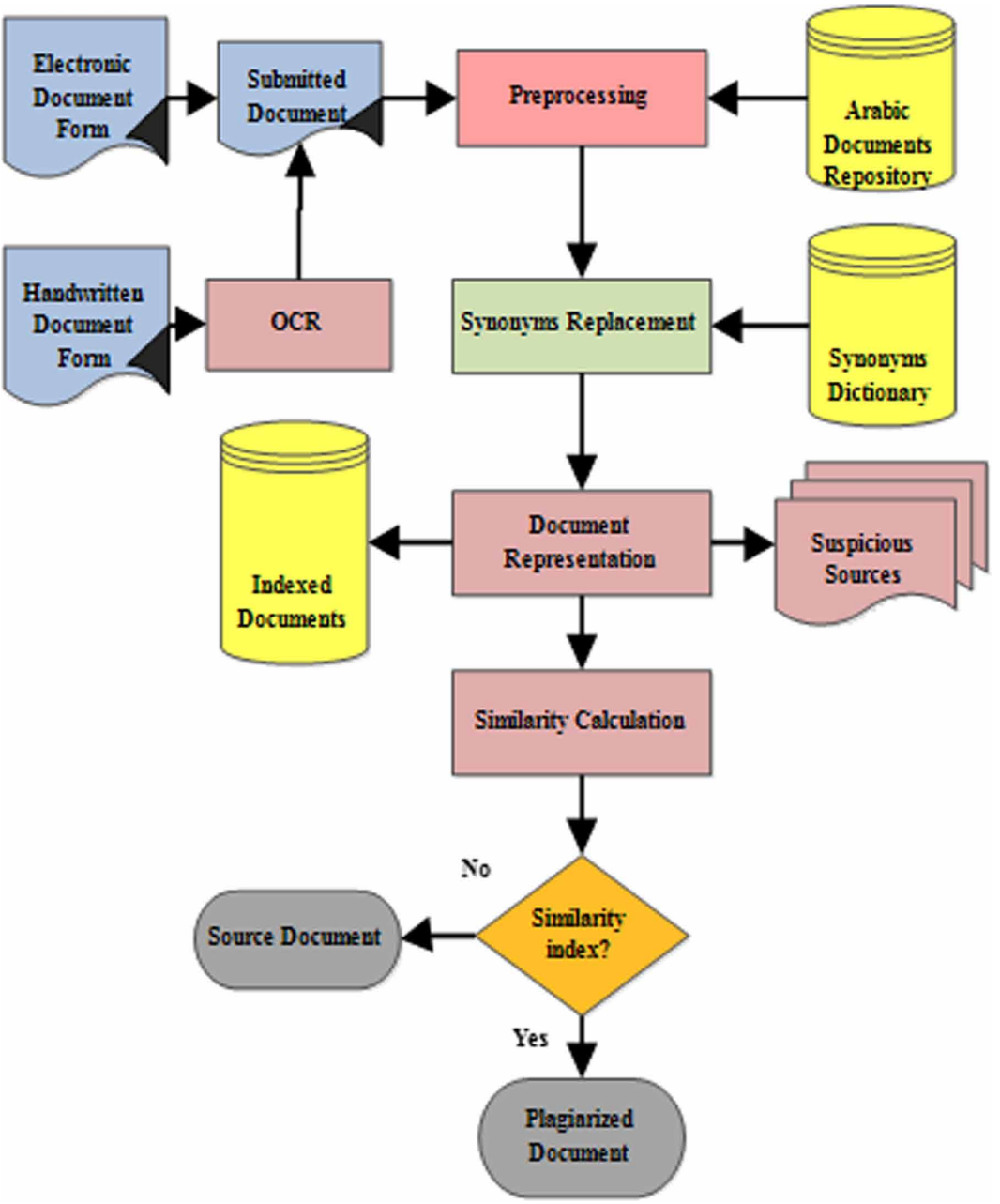
The source document retrieval module holds the user submitted document in an electronic form which contains text preprocessing module, query generation module and query submission module. The preprocessing module aims at doing three main tasks: 1-Tokenization, 2- Remove Stop Words and 3- Replacement of Synonym.

The framework of ASTAP is subdivided into three main modules as shown in Figure 2. First, document registration module (including submitted module, source document retrieval module, and Document Representation). Second, database module (including source documents collection, and web Saudi Digital Library (SDL) (Elhoseny et al., 2017). Third, similarity detection module (including Similarity Computation model and Similarity Report Generation module).

The Tokenization (Wang, et al., 2016) is responsible for breaking the stream of characters into tokens. Without recognizing the tokens, it is hard to see extracting higher_level info from the text. Each one is a kind of a type, so the number of tokens is much higher than the number of types. A computer software would catch the task more difficult. Therefore, after the user submits a document, the Tokenization module reads the file and breaks it down into tokens. The character space that we suggest is all the time delimiters and are not calculated as tokens.

A period, comma, or colon between numbers would not usually be considered a delimiter but rather part of the number. Any other commas or colons are delimiters and may be tokens. A period can

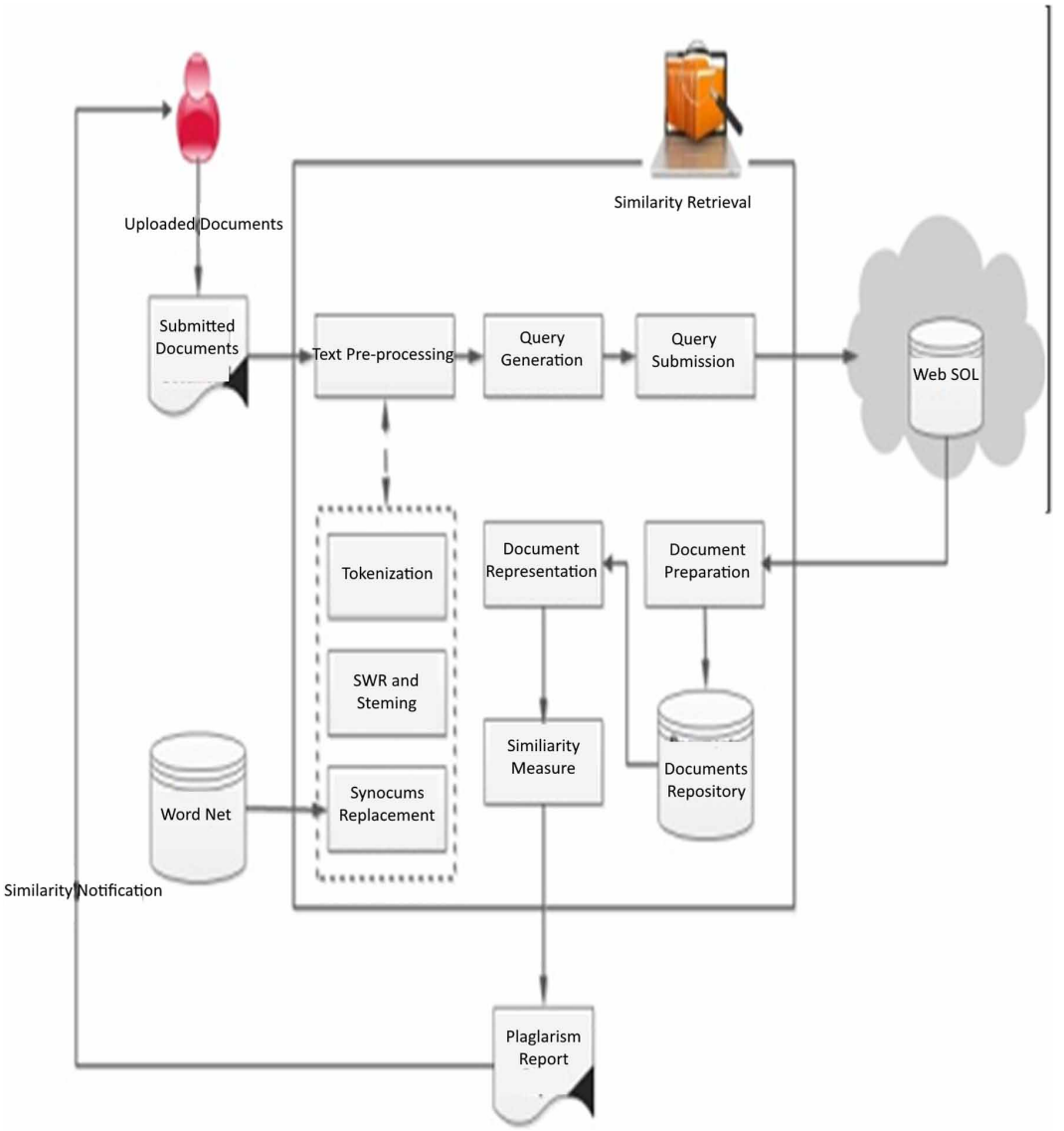
Figure 1. The ASTAP main operations



be part of an abbreviation when space follows it. However, some of these are the end of sentences. For tokenization, it is probably best to treat any ambiguous period as a word delimiter and also as a token.

The Stop_words_Removal and Rooting (Vani & Gupta, 2015) works on the raw manuscript to extract terms from text. Remove the non_informative text is usually uses a method in text recovery and classification. Stop words characterize the regularly happening, unimportant words that seem in a text file. Public stop words in English such as a, an, the, in, of, on, are, be, if, into, which, Whereas stop words in Arabic include: ") ,"الله" ,("الرحيم" ,"الرحمن" ,"الله" ,("بسم" . (من, إلى, عن, على, في

Figure 2. The ASTAP Framework



الذى", "لا",, "عبد", "عبيد", "هذا", "هذه", "هذان", "هتان", "هؤلاء", "انت", "انا", "نحن", "إني", "إنني", "انتما", "انتن", "أيا", "ايها", "ايتها", "كان", "امسى", "اصبح", "صار", "ليس, اضحى", "منذ", "ماذا", "لماذا", "عند", "متى", "كيف", "كيفما", "اين", "اينما", "الذي", "التي", "الذين", "اللائي", "اللاتي", "بما", "لمن", "لأن", ("لا", "اله", "الا", "الله, "النبي", ("القرآن", "الكريم"), "إياي", "إياك", "إيانا", "اياكما", "اياكن", "اياه", "اياها", "اياهما", "اياهم", "اياهن", "أيان", "اينما", "حيثما", "كأن", "ويكأن", "هيهات", "شتان", "سرعان", "مهما", "الى", "على", "إذا", "لولا", "عسى "لما", "ما زال", "ما دام", "بات", "ما برح", "ما انفك", "كاد ", "اوشك", ...etc. These words do not provide important meaning to the documents.

So, they should be deleted to minimize 'noise' and the calculation time. Some practitioners have felt that normalization more aggressive than stemming is advantageous for at least some text-preprocessing applications. These stemmers intend to reach a root form with no derivational prefixes and suffixes. For example, " سألتمونيها " is reduced to the stem " سأل ".

The final outcome of such violent stemming is to decrease the number of types in text group, thus creation distributional statics more consistent. Moreover, Word Stemming (Sindhu & Idicula, 2015) or Rooting: It will be altered to the word's basic shape. Firstly, the documents are broken down into words. Secondly, the words are characterized by their stems, for example, '' ,'يمشي المشي' and ' تمشي ' would be represented by the stem ' مشى '.

The group set of features is usually called a lexicon, and it works as an input a text token. Some rules must be defined such as token length, token content, token ends if the token has a special character and what to do if the token has a special word.

formerly reconnoitered

Using the Synonym (Sahi & Gupta, 2016) Replacement, the words are transformed to their most common alternative word which can assist to fined advanced forms of unseen plagiarism. The first alternative word in the list is considered as the most common one.

After processing the document, the query generation module starts its work to generate all possible parameter used for query submission module. Then, the query submission module sends all the query's parameters to SDL (Saudi Digital Library) to search the web for possible plagiarized documents. Dependently, the source document collection module downloads the searching results from SDL. The searching results are then prepared in the required file format, i.e., Docx, or PDF. The document representation component creates a document_tree_structure that defines its interior demonstration and filters the plagiarized source documents to save it in the repository. Using the similarity computation (Sharma & Jindal, 2016) module, tree definition (Thompson, et al., 2015) is made for each document to define its reasonable structure. Thus, the root defines the document himself, the next level defines the paragraphs, and the next nodes defines the sentences. It is intended to escape unimportant assessments among some documents. Trees are formerly reconnoitered top-down and compared first at the document level, then at the paragraph level and at the end at the sentence level. Finally, the similarity report generation module generates a report for the plagiarized documents including the sources and their URLs.

ASTAP is consists of three major modules, it can be classified into three major modules: Document registration module, database module, Similarity detection module.

1- Document Registration Module

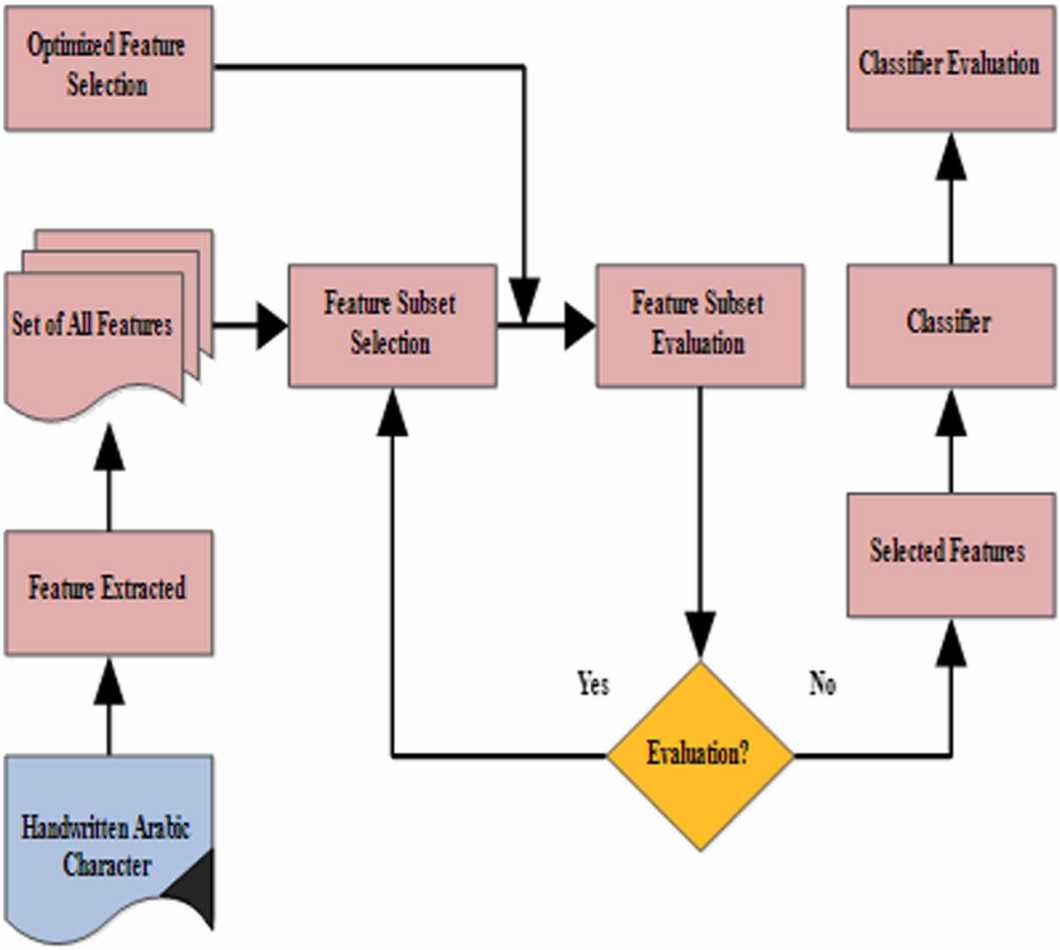
Document registration module or source document retrieval module used to preprocess the text and prepare it for similarity detection and add the document to the database. For a given input document, add The workflow in the database is:

(1) for a given document D, calculate the Df digital fingerprint;
(2) query the database whether there is the same digital fingerprint;
(3) If the document exists, skip this document and go to (i); otherwise, the system automatically generates a unique document id number Save, and then add the document id and the actual document name to the name mapping list;
(4) preprocessing the document to convert the digital documents of different formats into plain text formatted documents of uniform format;

2- Database Module

The system's database using MS-SQL Server database, the main table, including table_files and Table_text, respectively, the original document storage at the time of uploading the document and the contents of the preprocessed text. Which are included in the table table_filesId field, a file field to save the file name, a fingerprint field to save the text File generated by the digital fingerprint, and a URL field to save the document stored in the server-side physical path. table table_text include

Figure 3. The feature selection optimization model



txt_text Field, an XML._text field, a segment_text field, and a sentence_text field, the contents of which are explained below:

(1) txt_text field, pre-processed the original document text format make sure it is .doc
(2) xml_text field, XML formatted documents. XML format contains document properties (such as title, abstract, etc.) And document content.
(3) segment_text field, after the document word segmentation results. Content is a collection of ordered words, and mark each word The part of speech. In the word segmentation at the same time need to remove the text of the function words and stop words.
(4) sentence_text field, stored sentences that semantic meaning of verbs, nouns, and adjectives, as the digital fingerprint of the sentence.

3- Similarity Detection Module

It is the core module of the plagiarism detection system, the system goes through the document digital fingerprint comparison and determines whether or not the document was plagiarism detected before of the existing document in the database. Root (Val, P), where Val is the digital fingerprint of the entire document, P points to all Paragraph generated fingerprint tree; Paragraph Fingerprint tree node, Par (Val i, S i, C,) where Val i is the I-th paragraph number Fingerprints, S i is the i-th paragraph

all the sentences generate the fingerprint tree, C points to other nodes; the nodes of the fingerprint tree K i, Where Val i is the digital fingerprint of the i-th sentence, K i is the fingerprint tree of the corresponding text block of this sentence, and C is the other finger Chu(Val, N, V, A), where Val is the fingerprint of the text block and N points to the node of the text block, V is a collection of verbs, and A is a collection of adjectives. When detecting the matches of two documents, the fingerprints are first compared from each other, and if the fingerprints of the two documents are identical, the document to be tested is fully replicated. A paragraph with the same fingerprint records its position information, and the paragraph match counter is incremented by one. If the paragraph of the fingerprint is different, find the sentence the fingerprint tree of the child, and the sentence fingerprint tree, the same fingerprint of the sentence to write down its position information, the sentence match counter plus one. and find the sentence corresponding to the text block of the fingerprint tree, until the traversal of the entire tree when the matching task is completed,

METHODOLOGY

GWO-based OCR

The proposed OCR consists of three main phases. The pre-processing applies a set of operation on a raw image such as binarization and noise removing. Then, feature extraction and selection phase starts. This second phase is the core of the OCR functions which is implemented using GWO. As shown at Figure 3, in this phase, the most informative knowledge is extracted from a character image which helps us to recognize the characters in the document and selection of a relevant feature extraction algorithm is probably the most critical factor in achieving high recognition performance. Finally, the classification phase is essential before feeding to OCR as there is no universal OCR which recognizes multiple scripts. For that, we applied several machine learning techniques including Support Vector Machines (SVM) and Random forest (RF) (Poulos, 2016).

In this paper, GWO is designed to optimize/ reduce the feature subset; a solution represents the feature subset. In this section, we present basics of GWO algorithm and our proposed approach using GWO Algorithm together with SVM classifier to find the best combination of features. The dataset is separated into two parts, exercise and evaluation. The input is training dataset feature vectors with their corresponding classes and evaluation dataset in addition to the initialization of the parameters of both GWO and SVM, whereas the output is the optimal feature subset. Radial basis function (RBF) kernel function is selected and used in this paper, as it's the most sufficient for SVM.

Similarity Detection Using Modified AST

In different applications such as image and document processing applications (Yuan, et al., 2017) Abstract Syntax Tree (AST) is a similarity detection (Zaher, et al., 2017) algorithm that analyzes similar detection schemes to detect plagiarism efficiently. This paper proposes a modified version of AST. For each node, the modified AST determines a hash value (Hattab, 2015) and compares it with all other nodes. Also, the algorithm compares sub-trees based on the tree-structure with various reduction in its execution. ASTAP improves the way of syntax tree similarity and proposes a new procedure that reorders the nodes of ASTs to the longitudinal framework. The procedure consists of five main working steps. First, it generates AST. Second, it calculates the hash value (s). Third, it classifies the information of the node. Fourth, it compares these hash values. Finally, it calculates the degree of similarity. More details about the proposed procedure are presented in the next section.

ASTs Generation

ASTs generation is shown in Table 1. Initially, to produce Lexical Analyzer *LA*, we use l_{ex}, and to create Basics Analyzer *BA* (Arabic grammar called Basics) we use yacc. *BA* and *LA* act as a separate module to produce *ASTs*. After preprocessing, some changes are made on *BA* and *LA* to bring the symmetric *ASTs* groups *{ST}* and *{TT}* which have many of these changes. In our proposed system,

Table 1. Algorithm 1: AST generation algorithm

Algorithm 1. AST Generation Algorithm	
1	**Input**: D:= document text
2	**Output**: SD:= similarity degree
3	**Initialize**: // abstract class
4	**Set** Lex to Lexical analyzer LA
5	**Set** yacc to Basics analyzer BA //(Arabic grammar called Basics)
6	**Set** TT to target text {TT}
7	**Set** ST to suspect text {ST}
8	**Set** AST{TN} ∈ {TT}
9	**Set** AST{SN} ∈ {ST}
10	**Set** TMC:= count all nodes in {TT}
11	**Set** SMC:= count all nodes in {ST}
12	// **end class**

a whole *AST* for an Arabic document is produced from a class, an interface or a function. Suppose that the group all nodes of *AST{TN}* is included in *{TT}* and *{SN}* is included in *{ST}*. Assume that *TMC* is the total of all nodes of the *AST* in *{TT}*, and *SMC* is the total of all nodes of the *AST* in *{ST}*.

Hash Values Calculation

The hash values are calculated by going through all the *ASTs* within *{TT}* and determining a hash value for each node by collecting from end to top of the tree. In ASTAP, As seen in Figure 4, we suppose that the hash value of the subtree by node *X* as its top is known as *Hash(x)* (Jain & Kumar, 2016), the type hash value of *X is x*, and sub_nodes of *X* are $C1, C2, \ldots, Cn \in \{TN\}$ while $n \neq 0$, in which n happens $n \in N$. Formerly, we have an equation as follows:

$$Hash(x) = \begin{cases} X + \sum_{i=1}^{n} Hash(Ci), & 0 < n < TMC \\ X_1 & n = o \end{cases} \quad (1)$$

Also, for {ST} the Hash values of all the nodes are determined in the same way, as shown in Table 2. Where Hash(x) is the Hash value of the sub_tree whose root is *x; x* represents the Hash value of x's type and TMC is the total of all nodes of the AST with the all nodes in {TT}.

Node Information Classification

The working steps of a node information classification are shown in Table 3. For any node $X \in \{TN\}$, its actual information includes line start number *StNx*, hash value *Hx*, type of the node *NdNx* line end number EdNx and the number of sub-nodes *CCx*. They create the information vector.

Figure 4. Hash values calculation of an AST

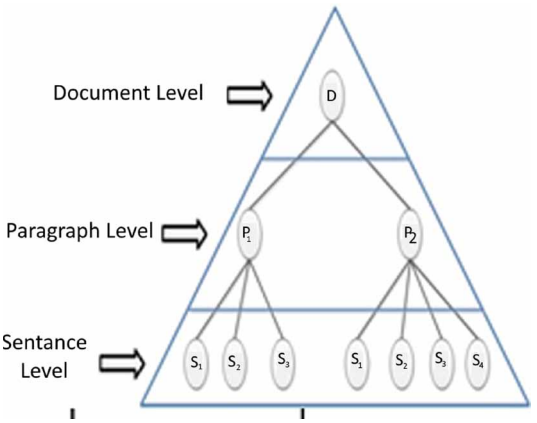


Table 2. Algorithm 2: Hash value calculation algorithm

	Algorithm 2: Hash Value Calculation Algorithm
1	**Input**: n, all AST in {TT}
2	**Output**: hs:= total hash value for {TT}
3	**Initialize**: ci:= node i, TMC:= count for all nodes, x:= node value
4	**for each** ci in {TT} **do**
5	**if** 0 < n < TMC **then**
6	ks:= x + ci
7	**Else**
8	ks:= x
9	**End if**
10	hs:= ks
11	**Endfor**
12	**return** hs

$$\text{INFO } x = (H, StN, EdN, NdN, CC) \tag{1}$$

After that, we make a group of the vector elements having the same number of sub_nodes,

$$INFOx \in \{INFON | N \in \{TN\}, CCN = CCx\} \tag{2}$$

Where *INFON* is the actual information node *N* vector and *CCN* is the node *N* number of sub-nodes. The real information in *{ST}* for all nodes is determined in the same way.

Hash Values Matching

In this phase, ASTAP goes through all nodes referred to their total of sub_nodes as illustrated in Table 4. For all nodes that achieve the case condition:

Table 3. Algorithm 3: Node Information classification algorithm

	Algorithm 3: Node information classification algorithm
1	**Input**: n, all AST in {TT}
2	**Output**: INFO (x):= Information for Node x in {TT}
3	**Initialize**: StNx = line start number, Hx = hash value for node x, NdNx = type of the node x.
4	EdNx = line end number for node x, CCx = number of sub-nodes
5	**For any node** X ∈ {TN}
6	INFO x = (H,StN,EdN,NdN,CC) x
7	INFOx ∈ {INFON | N ∈ {TN},CCN=CCx}
8	**return** INFO(x)

$$0 \leq i < \min\left(SMC, TMC\right)$$

Then the *H* values of all the components in

$$\{INFON \mid N \in \left\{TN\right\}, CCN = i\}$$

are compared with these in

$$\{INFON \mid N \in \left\{SN\right\}, CCN = i\}$$

Since the difference between the constructions of sub_trees and sub_node, we only match hash values of sub_trees with the same number of sub_nodes. Comparing all the nodes of *{TT}* with *{ST}*. For *ASTs,* the algorithm complexity is $O\left(n^2\right)$, where n is the minimum one through the whole amounts of *{T}* and *{S}*. Though, when classifying the nodes referred to their number of sub_nodes, the algorithm difficulty will down in the range of

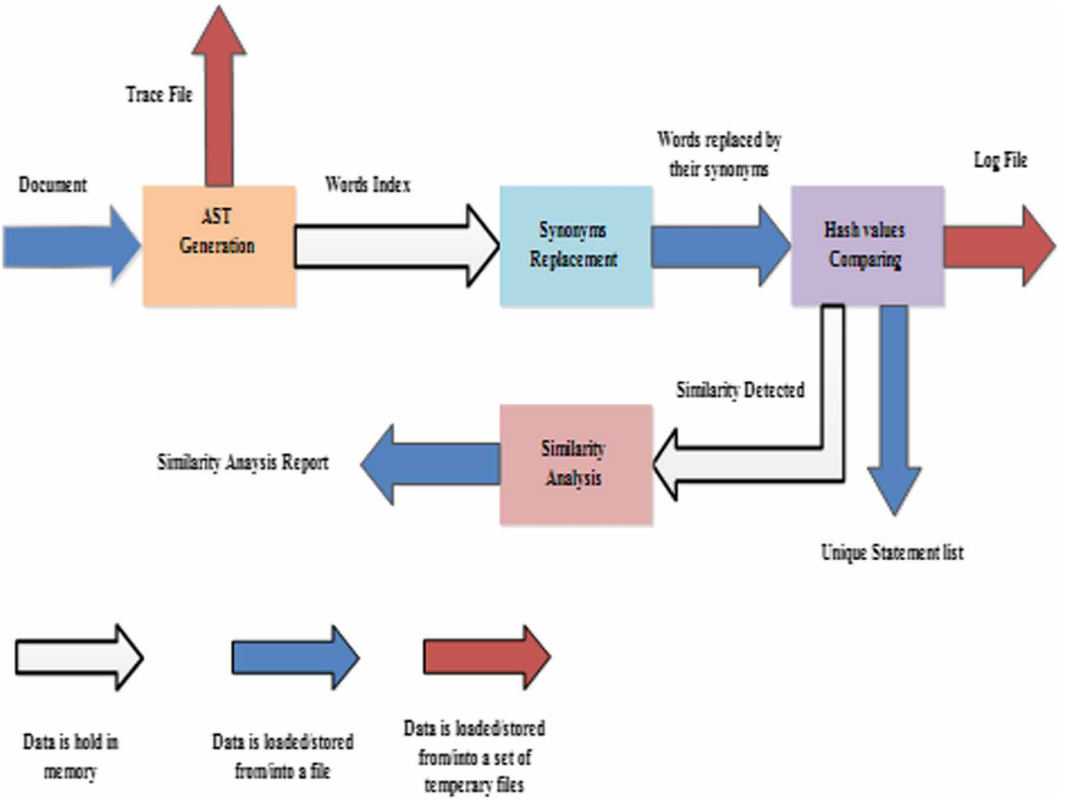
$$O\left(\frac{n^2}{MC}\right) and\, O\left(n^2\right)$$

where

MC=Min (SMC, TMC)

the minor one between *SMC* and *TMC*. This all shown in Figure 5 and Figure 6.

Figure 5. Hash values matching process



Similarity Calculation

Finally, the similarity calculation is shown in Table 5. After the Hash Values Matching (Bhushan,2015; Danti, 2015) process end, the similarity is calculated for all nodes that meet

$$INFOx \in \{INFON \mid N \in \{TN\}, CCN = i\} \tag{3}$$

and

$$INFOy \in \{INFON \mid N \in \{SN\}, CCN = i\} \tag{4}$$

if Hy = Hx, where *Hx* is component number one of *INFOx* and *Hy*, is component number one of *INFOy*. We consider the document crumbs symmetric when node *X* and node *Y* are similar. Then, we add *(INFOx, INFOy)* to a group and remove repeated information from the group that referred to the values of *StNx* and *EdNx* in *INFOx* to avoid redundancy calculations and makes a group of similar node information *SMINFO*. After that, the similarity degree SD (Elhoseny, et al., 2017) is calculated by using Eq. 2.

Figure 6. An example for hash values matching

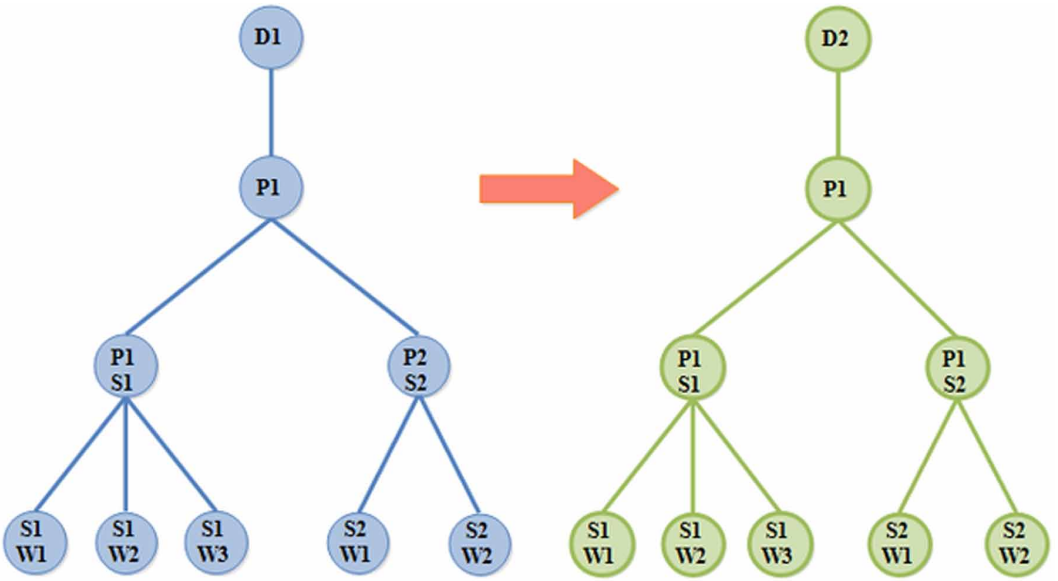


Table 4. Algorithm 4: Hash values comparing algorithm

Algorithm 4. Hash Values Comparing Algorithm	
1	Input: hs(i), hash value for node i
2	Output: H(i), hash value for compared node i
3	Initialize:
4	TMC = the total of all nodes of the AST in {TT}
5	SMC = the total of all nodes of the AST in {ST}
6	MC=min(SMC,TMC)
7	For any node i ∈ {TT}
8	if 0 ≤ i < MC then
9	compare H(i) in {INFON|N ∈ {SN},CCN=i} with
10	H(i) in {INFON|N ∈ {TN},CCN=i} endif
11	endfor
12	return H(i)

$$\text{SD} = \frac{\sum f\left(StNx\right)}{LC}$$

where, *X* node of the target of *AST*, across $\left(INFOx, INFOy\right) \in SMINFO$ and *LC* is the number of the lines in the target node.

As a final point, for all

$$\left(INFOx, INFOy\right) \in SMINFO\ \boldsymbol{V}$$

we calculate the number of similar nodes according to NdTx in INFOx.

Many similarity measures use fingerprint comparison, overall LCS (Longest Common Substring) and Levenshtein distance (LD). In the LD metric, it measures the smallest number of process: adding, deleting, or substituting to convert one txt to another. whereas, the LD between "Monday" and "Sunday" is four. The LCS depends on finding the longest substring which common in pair of texts. whereas, the longest common substring in "Monday" and "Sunday" is "nday." The ASTAP For all substrings, there is a hash value of length s of the pattern string and for all substrings of length s of the

Table 5. Algorithm 5: Similarity calculation algorithm

Algorithm 5. Similarity Calculation Algorithm	
1	Input: H(i), hash value for compared node i
2	Output: SD:= similarity degree
3	Initialize: Hx is component number one of INFOx
4	Hy is component number one of INFOy,
5	SMINFO a group of similar node information
6	For all nodes where
7	INFOx ∈ {INFON | N ∈ {TN},CCN=i} and
8	INFOy ∈ {INFON | N ∈ {SN},CCN=i}
9	if Hx = Hy then
10	INFO x = (H,StN,EdN,NdN,CC) x
11	INFO y = (H,StN,EdN,NdN,CC) y
12	SMINFO(INFO x, INFO y)
13	calculate SD(i) endif
14	endfor
15	SD= = sum (SD(i))
16	return SD

text string. After comparing the hash values of both the pattern and the text string. If the values of the hash for text and pattern are equal, then there is similarity between these text and pattern substrings.

A key issue in matching finding is to select the proper measure method. When detecting plagiarism, LCS and LD are more appropriate, because plagiarism includes of a text (Adding, deleting, etc.). Therefore, LCS have been used, since it is based on matches rather than distance.

RESULTS AND DISCUSSION

In this part, we discuss the experimental results of the suggested ASTAP system. The discussion is divided to two main parts. First, the experimental results and dataset of the proposed GWO-based OCR are described. Second, the results that are related to the similarity detection of Arabic documents using three different datasets are clarified.

The Results of GWO-based OCR

CENPARMI dataset (Shehab, et al., 2016)] is used in this work. The dataset contains about thousands of handwritten isolated Arabic character images. 328 writers wrote each. The samples were carefully selected to represent the 28 Arabic character forms (initial, medial, and final). In this paper, it was focused on the basic 28 characters.

The samples were divided into exercise and trying sets randomly. Three times of GWO search were done which produced three feature sets. Table 6 represents the best three sequences results for the well-known factor SVM.

In Table 6, the best three experiments by GWO and the whole feature set (SVM tested that) were chosen by adjusting carefully, the cost and the Gamma parameters. As seen, GWO achieved great accuracy in the range of 90.83% to 92.78% which is very acceptable rate. In addition, the time efficiency was between 1927 second to 8187 second. The complexity of the writing way greatly affects the required time to process the document. However, the accuracy is still close for all cases.

SIMILARITY DETECTION RESULTS

Evaluation Criteria and Datasets

The performance (Acampora & Cosma, 2015) of the suggested ASTAP is calculated using three different datasets (See Table 7). These datasets are formed using 60 Arabic documents. The datasets are gathered through extracting these documents from different Arabic resources available on SDL. Each dataset contains 20 different documents. Whereas the run of ASTAP and the other state-of-the-art systems are executed on processor Intel Core i5 with CPU speed of 2.4 GHz and 6 GB RAM with operating system Windows 7 Ultimate 64-bit. The description of these datasets is discussed below. The performance of the proposed ASTAP is calculated in different terms, such as the similarity, the

Table 6. The accuracy ratio and the period for the whole features set and for three best GWO rounds by SVM

Feature Set	Number of features (decrease)	SVM (Factors)	Accuracy Ratio (%)	Period (secs)
The total features set	717 (0%)	C=4.30e+02, G=0.003	92.78	8187
1st GWO round	242 (34%)	C=76.109, G=0.009	90.09	2641
2nd GWO round	265(37%)	C=76.109, G=0.008	91.37	1927
3rd GWO round	254 (35%)	C=1.81e+02, G=0.006	90.83	3211

precision, and the time efficiency using these different datasets. The similarity ratio is measured by searching for the matching contents of each document in the Datasets with those that are stored at SDL. A set of comparisons is conducted, as listed below, with the state-of-the-art methods.

Dataset 1: Synonym Replacement

Dataset 1 is prepared by selecting 20 documents from the list of documents that are stored on SDL. These selected documents are created by changing fifty percent (50%) of the whole sum of words randomly in each document with one of their synonyms. However, in Dataset 1 the Stop-words are not considered.

Dataset 2: Structure Change

To create the dataset 2, another 20 documents are generated from the documents that are stored on SDL by altering the structure of selected sentences randomly. The number of produced sentences represented fifty percent (50%) of the total number of sentences per each document.

Dataset 3: Hybrid

In addition to the 40 documents that are selected for Dataset 1 and Dataset 2, the remaining 20 documents are assigned to Dataset 3. The documents of Dataset 3 are formed by randomly copying words chosen with one of their replacements (twenty percent (20%) of the total number of words) and changing the structure of selected sentences (forty percent (40%) of the total number of sentences).

Results and Discussion

To measure the similarity ratio of ASTAP, Figure 7, Figure 8, and Figure 9 show the similarity ratio using Dataset 1, Dataset 2, and Dataset 3, respectively.

As shown in Figure 7, ASTAP detects the highest similarity ratio in each document. Regarding the second best method, ASTAP improves the similarity in the range of 10% and 43%. Turnitin yields the lowest similarity ratio. Despite APD is more recent than APlag and is designed for Arabic documents, it is shown that its results are not consistent.

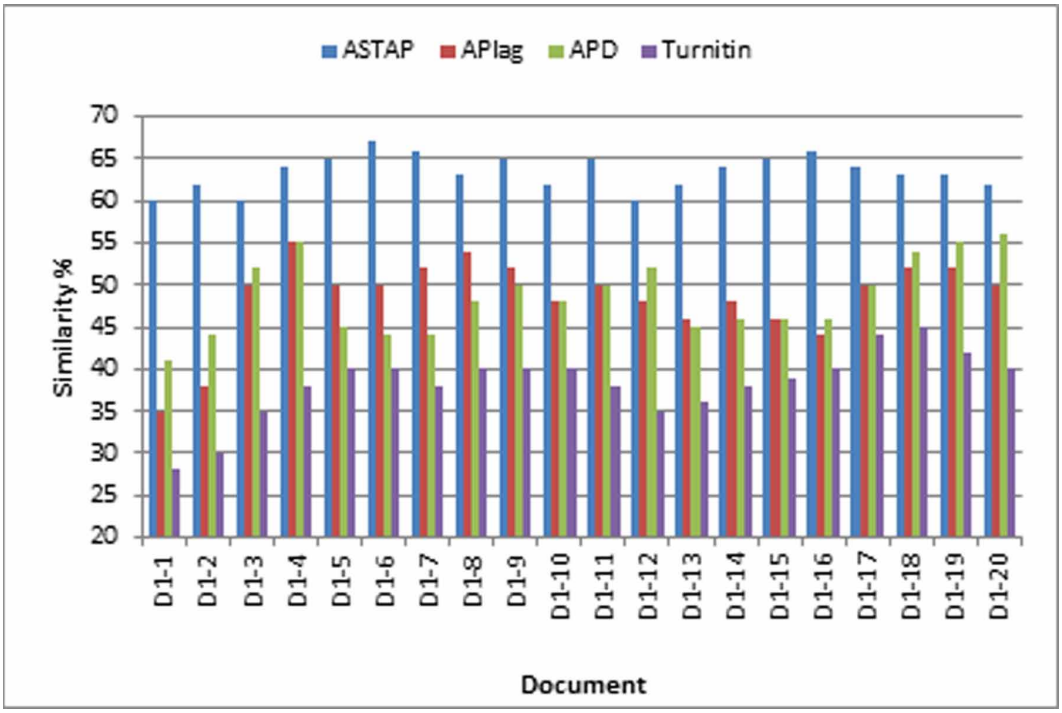
Contrary to Figure 7, the performances of all methods at Figure 8, which shows the similarity ratio using Dataset 2, are close. However, in most cases, ASTAP achieved the highest performance. Whereas, the highest percentage detected by ASTAP is 53% in D2-4. Consequently, Figure 9 reflects the performance of the ASTAP and the other state-of-the-art methods using Dataset 3.

However, the similarity cannot be used as the only performance measure criteria to evaluate the plagiarism detection systems. Therefore, the performance of ASTAP is measured using another important metric called Precision (Yalamanchili et al., 2016). The precision can be calculated as shown in Eq.1. It aims to evaluate the level of the credibility or the trustiness of the system.

Table 7. Statistics of the datasets

	Dataset 1	Dataset 2	Dataset 3
Average Count of Words/ Document	1029.6	1072.55	1054.6
Total Count of Images	94	75	91
Total Count of Tables	55	49	51
Synonyms Replacement Ratio	50%	0%	20%
Structure Change Ratio	0%	50%	0%

Figure 7. The similarity of ASTAP, Turnitin, APlag and APD using Dataset 1



$$Precision = \frac{\text{number of plagerized sequences identified}}{\text{total number of sequences determined}} \times 100$$

Dependently, the results of precision are listed in Table 8, Table 9 and Table 10 using Dataset 1, Dataset 2, and Dataset 3, respectively. Documents that are returned with a small percent of matches by less than 1% are excluded and counted as irrelevant documents. As shown in Table 8, Dataset 1 results indicate that the average precision of ASTAP is approximately 75%. The highest improvement of precision achieved by ASTAP regarding the second best method is 49%, which is shown at document D1-8. The reason for that is because most of the contents of document D1-8 are plaintexts. For that, the number of the identified plagiarized sequences is higher than the same number of the other documents.

The same was followed using Dataset 2. However, the overall precision ratio using Dataset 2 is less than the precision ratio for Dataset 1. As shown in Table 9, ASTAP achieves more stability in detecting the structure changes in all documents. The performance of all methods is close in that case due to the ignoring synonyms replacement.

Furthermore, Table 10 shows the precision results using Dataset 3. Since the documents at Dataset 3 are prepared using a hybrid method between the synonyms, which is used at Dataset 1, and the structure change, which is used at Dataset 2, the results show the balancing between the precision results that are obtained using Dataset 1 and Dataset 2.

Finally, the time efficiency (Elkhidir et al., 2015) is used as an essential factor to calculate the performance of the suggested method. Therefore, Table 11 compares the time efficiency of ASTAP with the different state-of-the-art methods. For each Dataset, the maximum, the minimum, and the average time are calculated.

Figure 8. The similarity of ASTAP, Turnitin, APIag and APD using Dataset 2

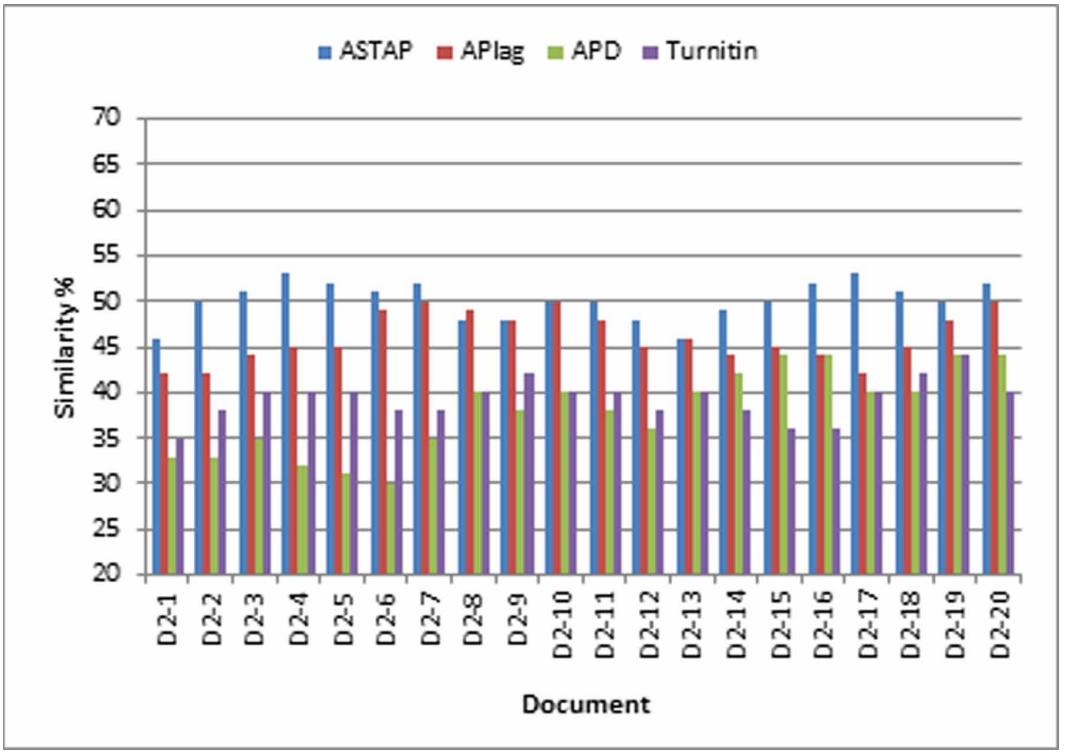


Figure 9. The similarity of ASTAP, Turnitin, APIag and APD using Dataset 3

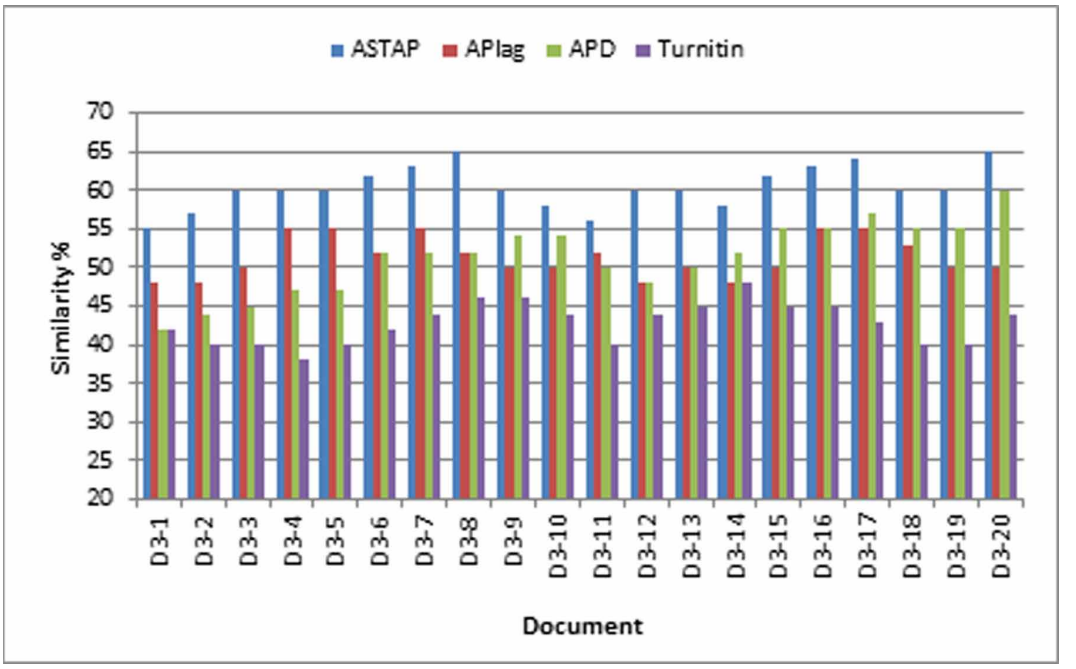


Table 8. The precision of ASTAP, Turnitin, APlag and APD using Dataset 1

	ASTAP	APlag	APD	Turnitin
D1-01	70	40	38	25
D1-02	72	42	44	28
D1-03	75	49	55	35
D1-04	74	35	51	29
D1-05	75	40	55	31
D1-06	79	43	58	33
D1-07	85	50	62	38
D1-08	88	55	59	38
D1-09	78	57	58	35
D1_10	75	55	55	29
D1_11	76	44	60	35
D1_12	69	53	62	29
D1_13	65	56	55	31
D1_14	68	44	50	33
D1_15	71	42	44	36
D1_16	73	47	50	32
D1_17	72	45	52	31
D1_18	77	48	54	35
D1_19	80	49	51	37
D1_20	81	50	56	30

The results of the time efficiency indicate that the max processing time required by ASTAP using Dataset 1 is 122 which is still shorter than the necessary processing time that is consumed by any other method. The longer running time is obtained in D1-8 which contains, the larger number of plaintexts. By avoiding the complexity of APlag and APD in document processing, AST is highly simplified the tokenization, the stopword removal, and rooting processes, which make the proposed method faster than the other method in all Datasets.

CONCLUSION AND FUTURE WORK

In Internet-based documents, processing handwritten documents for mining their contents is considered as one of the most difficult challenges. Therefore, the process of plagiarism detection in such documents, especially in handwritten Arabic documents, is an urgent need. Dependently, this paper presents an intelligent model for a plagiarism detection system called ASTAP which is used to detect some unobserved forms of plagiarism such as synonym alternation or change of sentence structure in such complicated multimedia documents. There are two main contributions of that paper. First, a handwritten Arabic character recognition system is proposed using Grey Wolf Optimization (GWO) algorithm. Second, a modified Abstract Syntax Tree (AST) is used to match the contents of the Arabic documents to detect any similarity. Compared to the state-of-the-art methods, ASTAP improves the effectiveness of plagiarism detection in handwritten Arabic documents regarding the matched similarity ratio, the precision ratio, and the processing time. Regarding the second-best

Table 9. The precision of ASTAP, Turnitin, APlag and APD using Dataset 2

	ASTAP	APlag	APD	Turnitin
D2-01	50	48	43	40
D2-02	52	47	45	42
D2-03	53	45	42	40
D2-04	49	45	41	42
D2-05	49	44	44	40
D2-06	51	43	45	45
D2-07	49	45	46	42
D2-08	52	46	46	40
D2-09	50	45	45	42
D2-10	53	45	41	40
D2_11	52	42	46	45
D2_12	54	45	44	41
D2_13	55	48	45	43
D2_14	51	45	46	42
D2_15	49	44	42	40
D2_16	49	45	41	39
D2_17	51	44	43	38
D2_18	53	45	44	40
D2_19	52	47	42	42
D2_20	53	44	45	40

method, ASTAP improves the detected similarity ratio in a range of 10% and 43%, while it enhances the precision in a variety of 22% and 49%. Besides, it reduces the running time by 12.5%. Also, the proposed GWO-based OCR achieved great accuracy in the range of 90.83% to 92.78%. In addition, its time efficiency was between 1927 second to 8187 seconds.

This result shows, ASTAP, a system that works on the Internet to enable specialists to detect thefts of electronic texts in Arabic so it can be integrated with e-learning systems to ensure the safety of students and research papers and scientific theses of electronic thefts. It also describes the major components of this system, including stage outfitted, and in the end, we will establish an experimental system on a set of documents and Arabic texts and compared the results obtained with some of the existing systems, particularly TurnItIn.

The future work will concentrate on improving and adding more choices in this tool. Most essentially is to test our ASTAP system in various universities.

Table 10. The precision of ASTAP, Turnitin, APlag and APD using Dataset 3

	ASTAP	APlag	APD	Turnitin
D3-01	55	45	45	38
D3-02	57	47	50	40
D3-03	60	47	52	41
D3-04	62	50	54	44
D3-05	65	52	50	45
D3-06	62	54	48	44
D3-07	60	51	45	40
D3-08	58	50	44	38
D3-09	60	48	44	36
D3_10	64	50	43	39
D3_11	66	48	45	40
D3_12	65	46	40	40
D3_13	62	45	44	38
D3_14	60	47	42	36
D3_15	58	49	45	38
D3_16	62	51	42	40
D3_17	64	49	40	43
D3_18	60	50	41	40
D3_19	62	52	45	38
D3_20	64	50	44	40

Table 11. Time efficiency of ASTAP compared to the state-of-the art methods

		ASTAP	APlag	APD	Turnitin
Dataset 1	Max.	122	132	133	135
	Min.	109	117	122	123
	Avg.	115.3	125.85	126.5	128.75
Dataset 2	Max.	125	130	132	134
	Min.	120	123	125	129
	Avg.	122.7	127.1	129.05	130.85
Dataset 3	Max.	126	130	133	135
	Min.	117	122	127	129
	Avg.	122.2	127	129.6	132.55

REFERENCES

Abdelrahman, Y., Khalid, A., & Osman, I. (2014). A survey of plagiarism detection for Arabic documents. *International Journal of Advancements in Computing Technology*, *4*(6), 34–38.

Abdi, A., Idris, N., Alguliyev, R., & Aliguliyev, R. M. (2015). PDLK: Plagiarism detection using linguistic knowledge. Science Direct. *Expert Systems with Applications*, *42*(1), 8936–8946. doi:10.1016/j.eswa.2015.07.048

Acampora, G., & Cosma, G. (2015). A Fuzzy-based approach to programming language independent source-code plagiarism detection. *Journal of Digital Information Management*, *14*(2), 124–135.

Ahtiainen, A., Surakka, S., & Rahikainen, M. (2011). Plaggie: Gnulicensed source code plagiarism detection engine for java exercises. In *Proceedings of the 6th Baltic Sea Conference on Computing Education Research: Koli Baltic* (pp. 141–142). Academic Press.

Aiken, A. (2015). Moss: A system for detecting software plagiarism. *University of California–Berkeley*, *23*(4), 245–259.

Alzahrani, S., & Salim, N. (2015). Statement-based fuzzy-set IR versus fingerprints matching for plagiarism detection in Arabic documents. In *Proc. of the 5th Postgraduate Annual Research Seminar (PARS09)*, Johor Bahru, Malaysia (pp. 33-49). Academic Press.

Alzahrani, S. M., & Salim, N. (2008). Plagiarism detection in Arabic scripts using fuzzy information retrieval. In *Proceedings of the 2008 Student Conference on Research and Development (SCOReD 2008)*, Johor, Malaysia (pp. 112-119). Academic Press.

Bhushan, S. B., Danti, A., & Fernandes, S. L. (2017). A novel integer representation-based approach for classification of text documents. In *Proceedings of the International Conference on Data Engineering and Communication Technology* (pp. 557-564). Springer Singapore.

Borner, K., Chen, C., & Boyack, K. (2012). Knowledge Domain Visualization. *Information Visualization*.

Chen, J. Y., & Zhao, C. Z. (2017). Tianji: Implementation of an Efficient Tracking Engine in the Mobile Internet Era. *IEEE Access*, *5*(1), 16592–16600. doi:10.1109/ACCESS.2017.2736064

Chow, K., & Salim, N. (2013). Web based cross language plagiarism detection. In *Proceedings of the Second International Conference on Computational Intelligence, Modeling and Simulation journal of computing* (pp. 199–204). Academic Press.

Clough, P. (2014). *Plagiarism in Natural and Programming Languages: An Overview of Current Tools and Technologies*. University of Sheffield.

El Bachir, M., & Bagais, M. (2014). APlag: A Plagiarism Checker for Arabic Texts. *I.J. Information Technology and Computer Science*, *10*(2), 80–89.

Elhoseny, M., Metawa, N., & Hassanien, A. (2017). Intelligent Information System to Ensure Quality in Higher Education Institutions, Towards an Automated E-University. *International Journal of Computational Intelligence Studies*, *6*(2), 115–149. doi:10.1504/IJCISTUDIES.2017.089049

Elhoseny, M., Zaher, M., & Shehab, A. (2017). FPSS: Fingerprint-Based Semantic Similarity Detection in Big Data Environment. In *Proceedings of the 8th IEEE International Conference on Intelligent Computing and Information Systems (ICICIS)* (pp. 1221-1233). IEEE.

Elkhidir, M., Ibrahim, M., & Awadalla, M. (2015). Plagiarism detection using free-text fingerprint analysis. In *Proceedings of the World Symposium on Computer Networks and Information Security (WSCNIS)* (pp. 213-223). Academic Press.

Grozea, C., & Popescu, M. (2011). ENCOPLOT: Pairwise Sequence Matching in Linear Time Applied to Plagiarism Detection. In *Proceedings of the 25th Annual Conference of the Spanish Society for Natural Language Processing, SEPLN* (pp. 10–18). Academic Press.

Hattab, E. (2015). Cross-Language Plagiarism Detection Method: Arabic vs. English. In *Proceedings of the International Conference on Developments of E-Systems Engineering (DeSE)*. Academic Press. doi:10.1109/DeSE.2015.25

Hussein, A. (2016). Visualizing Document Similarity Using N-Grams and Latent Semantic Analysis. In *Proceedings of the SAI Computing Conference*, London (pp. 269-279). Academic Press. doi:10.1109/SAI.2016.7555994

Jain, V. K., & Kumar, S. (2016). Extraction of emotions from multilingual text using intelligent text processing and computational linguistics. *Journal of Computational Science*, *21*(3), 316–326.

Kahloula, B., & Berri, J. (2016). Plagiarism detection in arabic documents: approaches, architecture and systems. *Journal of Digital Information Management*, *14*(2), 124–135.

Lam, S., Lee, K., & Choi, S. (2016). iChecker: An efficient plagiarism detection tool for learning management systems. *International Journal of Systems and Service-Oriented Engineering*, *3*(6), 16–31.

Mozgovoy, M., & Frederiksson, K. (2014). Fast Plagiarism Detection System, String Processing and Information Retrieval. In *Proceedings of the 12th International Conference (SPIRE 2014)* (pp. 267–270). Academic Press.

Osman, A., Salim, N., Binwahlan, M., Alteeb, R., & Abuobieda, A. (2012). An improved plagiarism detection scheme based on semantic role labeling. *Applied Soft Computing*, *12*(2), 1493–1502. doi:10.1016/j.asoc.2011.12.021

Poulos, M. (2016). Near duplicate text detection using graph depiction. In *Proceedings of the 7th International Conference on Information, Intelligence, Systems & Applications (IISA)* (pp. 121-127). Academic Press. doi:10.1109/IISA.2016.7785368

Sahi, M., & Gupta, V. (2016). Efficiency comparison of various plagiarism detection techniques. In *Proceedings of the International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)* (pp. 161–172). Academic Press. doi:10.1109/ICEEOT.2016.7755245

Seyedali, M., Seyed, M., & Andrew, L. (2014). Grey wolf optimizer. *Journal of Advances in Engineering Software*, *69*(7), 46–61.

Sharma, K., & Jindal, L. (2016). An improved online plagiarism detection approach for semantic analysis using custom search engine. In *Proceedings of the 3rd International Conference on Computing for Sustainable Global Development (INDIACom)* (pp. 124-129). Academic Press.

Shehab, S., Elhoseny, M., & Hassanien, A. (2016). A hybrid scheme for Automated Essay Grading based on LVQ and NLP techniques. In *Proceedings of 12th International Computer Engineering Conference (ICENCO)* (pp. 65-70). Academic Press. doi:10.1109/ICENCO.2016.7856447

Si, A., Leong, H., & Lau, R. (2012). CHECK a document plagiarism detection system. In *Proceedings of ACM Symposium for Applied Computing* (pp. 70-77). Academic Press.

Sindhu, L., & Idicula, S. (2015). Fingerprinting based detection system for identifying plagiarism in Malayalam text documents. In *Proceedings of the International Conference on Computing and Network Communications (CoCoNet)* (pp. 553-558). IEEE.

Subba, L. (2014). An anti-plagiarism add-on for web-CAT [Doctoral dissertation]. National University of Ireland Maynooth.

Thompson, V., Panchev, C., & Oakes, M. (2015). Performance evaluation of similarity measures on similar and dissimilar text retrieval. In *Proceedings of the 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K)* (pp. 229-234). Academic Press.

Vani, K., & Gupta, D. (2015). Investigating the impact of combined similarity metrics and POS tagging in extrinsic text plagiarism detection system, International Conference on Advances in Computing *Communication Information*, *11*(9), 727–738.

Wang, X., Evanini, K., & Mulholland, K. (2016). Automatic plagiarism detection for spoken responses in an assessment of English language proficiency. In *Proceedings of the Ninth Workshop on Innovative Use of NLP for Building Educational Applications* (pp. 22–27). Academic Press. doi:10.1109/SLT.2016.7846254

Wise, M. (2012). Yap3: Improved detection of similarities in computer program and other texts. *ACM SIGCSE Bulletin*, *28*(1), 130–134.

Yalamanchili, J., Green, R., & Xu, K. (2016). Performance enhanced multiset similarity joins. *Computer Communications*, *15*(1), 21–28.

Yuan, X., Elhoseny, M., & Riad, A. (2017). A genetic algorithm-based, dynamic clustering method towards improved WSN longevity. *Journal of Network and Systems Management*, *25*(1), 21–46.

Zaher, M., Shehab, A., Elhoseny, M., & Osman, L. (2017, September). A New Model for Detecting Similarity in Arabic Documents. In *Proceedings of the International Conference on Advanced Intelligent Systems and Informatics* (pp. 488-499). Cham: Springer.