A NOVEL APPROACH FOR RECOGNIZING TEXT IN ARABIC ANCIENT MANUSCRIPTS

Ashraf Nijim¹, Ayman El Shenawy², Muhammad T. Mostafa³ and Reda Abo Alez⁴

¹,²,⁴Computers and Systems Engineering Department, Faculty of Engineering, Al-Azhar University, Cairo, Egypt
³Mathematics Department, Faculty of Science, Al-Azhar University Cairo, Egypt

ABSTRACT

In this paper a system for recognizing Arabic ancient manuscripts is presented. The system has been divided into four parts. The first part is the image pre-processing where the text in the Arabic ancient manuscript will be recognized as a collection of Arabic characters through three phases of processing. The second part is the Arabic text analysis which consists of lexical analyzer; syntax analyzer; and semantic analyzer. The output of this subsystem is an XML file format that represents the ancient manuscript text. The third part is the intermediate text generation, in this part an intermediate presentation of the Arabic text is generated from the XML text file. The fourth part of the system is the Arabic text generation, which converts the generated text to a modern standard Arabic (MSA) language (this part has four phases: text organizer; pre-optimizer; semantics generator; and post-optimizer).

KEYWORDS

Ancient Manuscripts, Arabic Text Recognition System, Arabic Text Analysis, Arabic Text Generation.

1. INTRODUCTION

Libraries over the world have a huge amount of ancient Arabic manuscripts that have been kept in safe places. Those manuscripts are subject to many aging artifacts; natural disasters; theft or damage. Many of those manuscripts have not been recently published or even have an electronic copy. For those reasons and in order to benefit from the Arabic ancient library, it becomes a necessity to create a system that not only converts those manuscripts into an electronic version that researchers and concerned people can read, but also converts the text included in these manuscripts into modern, readable, and understandable forms.

Arabic ancient manuscripts carry the civilization and the religions of the Arabic area from the different historical ages. The Arabic ancient manuscripts hold all the different writing styles of the Arabic language. After the appearance of Islam in the Arabic area, many copies of the holy Koran were written and sent to different cities of the Middle East and North Africa. Different Arabic writing styles appeared in the newly converting to Islam cities. Figure 1 shows a sample of an Arabic ancient manuscript [1].

The required work for developing a system like this includes two primary factors: text recognition, and text analysis. The text recognition, manipulates the ancient manuscript as an image that must be converted to a set of characters and sentences. Due to the reason that these manuscripts had been written in different periods of time through the Islamic history, also, these
manuscripts had been written by people belonged to different countries and therefore, different cultures which make it even more diverse and more complex to understand. The importance of the second factor has been represented in translating the generated characters and sentences into another easier and understandable form to the current researchers and readers.

Al–Khalil is a project for building an Arabic infrastructure ontology system [2]. The project aimed to build an ontology centred infrastructure for Arabic resources and applications.

Arabic WordNet (AWN) is an expending project for Arabic words, their properties and relations. AWN is an extension to the WordNet (WN) project [3]. WN is a machine-readable lexical database that groups words into clusters of synonyms called synsets. Every synset can be thought of as a representation of a unique word sense meaning or concept. Currently AWN consists of 11,270 synsets, and contains 23,496 Arabic expression words and multiword [4].

In[5]authors present an Arabic Computational Morphology system. Their approach is a computational approach to Arabic morphology description that is drawn from lexeme-based morphology. The priority is given to stems and granting a subordinate status to inflectional prefixes and suffixes. In [6] authors present a Memory-based Morphological Analysis, and Part-of-speech Tagging of Arabic system. The system was based on data from the Arabic Treebank. Morphological analysis is performed as a letter-by-letter classification task. Classification is performed by the k-nearest neighbour algorithm. Part-of-speech tagging is carried out separately from the morphological analyzer. A memory-based modular tagger is developed with a sub-tagger for known words and one for unknown words.

The remainder of this paper is organized as follows; the next section is a presentation of our Arabic text recognition system and its four parts. Then our conclusion and future work.

2. ARABIC TEXT RECOGNITION SYSTEM

The Arabic ancient manuscripts recognition system is composed of three main sub-systems: manuscripts recognition; Arabic text analysis; and Modern Standard Arabic (MSA) text generation. Figure 2 shows those three sub-systems and the different steps of each subsystem.
2.1. Manuscript Recognition

The manuscript recognition sub-system deals with the manuscript as an input image and produces a series of Arabic character shapes that represent the handwritten Arabic text inside the manuscript; also it extracts the formats of the original manuscript such as the number of paragraphs, margins, and text size. Those formats are to be used in presenting the final output of the system. The manuscript recognition subsystem consists of four steps as indicated in Figure 3.

![Diagram of manuscript recognition system](image)

Figure 2. Arabic ancient manuscripts text recognition system main parts and subsystems.

2.1.1 Manuscript Binarization

The binarization step converts the manuscript image into black text pixels representing the text and a white background, a special binarization algorithm was designed to deal with different aging, storage and weather effects of the ancient manuscripts in general [7].

2.1.2 Manuscript Segmentation

The Arabic handwritten segmentation is considered as a difficult problem because of the high variability of the Arabic semi-cursive script language. An algorithm for performing manuscript segmentation was developed so that it does not look for words but instead it deals with the segmentation problem as a collection of Parts-of-Arabic Words (PAWs) [8]. This method gives a good result for segmentation of the handwritten Arabic text and avoids most of the segmentation known problems [8]. Figure 4 shows the different steps of the segmentation process.
In the pre-processing step, the white margins of the manuscript were removed using vertical and horizontal histograms of the manuscript image. Figure 5 shows the results of removing the margins of a manuscript. After that, skew detection was applied to detect any slant in the handwritten text. A slant correction method was applied to each text area inside the manuscript to fix the skew. Projection profiles were used to find the slant angle. The angle with the maximum variance of the black pixels is chosen to be the slant angle [9].

The coordinate rotation transformation method was used to correct the angle of the text. At the end of this pre-processing step a salt and pepper noise remove is applied to remove any artifact caused by the previous pre-processing methods. Figure 6 shows the results of the skew correction of an Arabic text.
The second step is the segmentation of lines using projection profiles (PP). PPs are used to find the maximum variances of black pixels inside the text area as it passes between text lines [3], [4], [5]. Figure 7 shows the result of segmenting two Arabic lines.

In the third step, the generated text lines are divided into part of Arabic words (PAWs). The vertical histogram of each line was used to identify between PAW areas [3]. Figure 8 presents an example of the segmentation results of a handwritten line into PAWs.

In the last step of the segmentation process the generated PAWs was segmented into simple-to-process characters. The Connected Components (CC) method was applied to identify the different parts of the PAW. The largest CC is considered the main object of the PAW. After that, the baseline detection method of main object was applied to detect the base line of the PAW. Figure 9 shows the output of the CC method. Relatively small CCs above and below the detected base-line of the PAW are considered as points, diacritics or Hamza (refers to Arabic alphabet character (ء)). More details about base-line detection method are available in [2] [3] [5] [6].
The segmentation of words to individual characters method in [10] was used to divide the PAWs into a collection of characters according to the calculated width and the vertical projection histogram of the middle zone for each PAW. The input and the output of this method were shown in fig. 10.

(a) (b) (c) (d)

Figure 10. (a) CC of a PAW, (b), (c) and (d) segmentation results of (a)

2.1.3 Features Extraction

A new enhanced edge detection method based on cooperation between edge algorithms was developed to improve the results of the SIFT and holes features extraction of our system [11].

Nine types of features were used in our system to extract all the varieties of the ancient handwritten characters. Figure 11 shows the main steps of the features extraction process, and Figure 12 shows the different steps of the feature extraction process.

Character image

Pre-processing

Features extraction

Character features vector

Figure 11. Features extraction process

Each entry undergoes a pre-processing before the features extraction. Thinning is one of the main pre-processing used in our system. This process simplifies the features extraction process and makes it faster. Following, we will describe each of those features in details.
Central Hu-Moments: the first four central Hu-moments of the character are parts of the features. Those moments are invariant to scaling; orientation; and position changing [11].

SIFT: is a feature vector of four numbers that represents the location coordinates \((x, y)\), scale, and orientation of the feature inside the character image. A maximum of 10 SIFT features for each character image are used in our system [12][13] [14].


Center of Mass: is a geometric feature of the character [14]. The computation of center of mass was done on the character before applying the thinning process. The distribution of black pixels of the character is balanced around this point.

Pattern Dimension Ratio: the value of the character image (CI) width divided by the height of the black pixels boundary box [14].

Holes: used to record location and dimension of all the founded holes in the CI [8]. This can be done by dividing the CI into four by four numbered boxes and record the box number that have the center of the hole as a feature. Also, the relative diameter of the hole is computed and recorded as a feature. Figure 13 shows the extraction of a hole from a PAW “Fi”.

Black Ink Histogram: each CI has two black ink histograms vertical and horizontal [14] by normalizing the CI to a fixed size of 15X10 pixels (see Figure 14). The twenty five histogram values of both the vertical and the horizontal directions are added to the features vector of the CI.

Hough transform is used to process the skeleton of CI so that the line parameters are added to the features vectors if the length of the line exceeds a relative threshold [9].

Corners: four different 4X4 templates are used to extract corners of characters [15] [16][17]. Then the CI is divided into three vertical areas (top) and three horizontal areas (right). These templates and the result of the corner feature extraction are presented in fig. 15.
### 2.1.4 Classification

Classification is the final step of the proposed system, and it includes two steps: learning and running operations. In the learning operation, the system is fed by a collection of CI’s features vectors along with their belonging classes. The system is trained to identify those different characters and connect each with its relative class using features. The learning operation is usually done once until a satisfied result is reached. The testing of this operation is done on another collection – other than those used in the learning part – of CI’s features vectors. This process is continued until satisfactory results have been reached.

![Figure 15. (a) Four different templates (b) result (c) horizontal and vertical areas.](image)

In the running operation, the system can recognise the class to which the entered CI’s belongs to, according to the training operation. The system efficiency depends largely on the training operation.

Two well known classifiers have been used during our experiments, artificial neural network (ANN) [9] and support vector machine (SVM) [9][18][19]. Although ANN is considered one of the oldest and successful classifiers, it has a remarkable low speed performance than SVM [9][18][19]. SVM is more efficient due to the use of sequential minimal optimization (SMO) training techniques.

A comparison was conducted between ANN and SVM using a sample of 20 Arabic handwritten numbers categorized from 0 to 9. The results of this comparison were showed in Table 1. These results indicate that SVM is more efficient and takes less run time than ANN classifier.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Model Run time (seconds)</th>
<th>% of correctly classified instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neural Network (NN)</td>
<td>6.02</td>
<td>95</td>
</tr>
<tr>
<td>Support Vector Machine (SVM) – SMO</td>
<td>1.6</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 2 indicates the run time and the accuracy for training set of 3800 characters and testing of other 380 characters.
Table 2. SVM-SMO Classifiers Results for the Test Set

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Model Run time (seconds)</th>
<th>% of correctly classified instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support Vector Machine (SVM) - SMO</td>
<td>32.6</td>
<td>71</td>
</tr>
</tbody>
</table>

The ancient Arabic manuscripts recognition system gave promising results. The unavailability of a database of Arabic ancient manuscripts or Arabic ancient words made the comparison of any available recognition system impossible. Furthermore, all the research done so far on Arabic ancient handwritten scripts focused on one of the important phases mentioned before such as segmentation or binarization.

2.2 Arabic Text Analysis

The Arabic text analysis is the second part of the ancient Arabic text recognition system. This part takes the output of the image recognition part in a form of a text document. It consists of three phases: the lexical analyzer, the syntax analyzer, and the semantics analyzer.

2.2.1 Lexical Analyzer

The lexical analyzer phase has two components: Arabic text segmentation (cuts the input text document into different Arabic POS), and the POS checking (spelling check and suggestions for incorrect spelled POS are presented to choose from).

Arabic Text Segmentation: The input to this component consists of a set of paragraphs each paragraph consists of a set of sentences. These paragraphs will be divided into individual sentences as described in Fig.16.

Input: Text document
Output: a set of sentences and paragraphs that constitute the input text document.
Process:
Step 1: Search for spaces between characters and mark the group of characters between any two spaces as POS.
Step 2: Mark the start of each paragraph as a start of new sentence.
Step 3: Search for the symbols (., ;, ?, !) and mark its position as end of sentence.
Step 4: Mark the next position to the end of sentence as a start a new sentence.
Step 5: If the starts of a paragraph or a sentence mark the characters between the first paragraph or sentence and the space as POS.

Figure 16. Arabic Text Segmentation Algorithm.

Arabic POS Checking: The individual POS results of the previous phase are checked for completeness. Missing parts would be identified and completed by referring to the Arabic dictionary[20]. Different suggestions may be provided for a single POS. Each of those is then tagged according to its type [21][22]. Figure 17 presents the Arabic POS missing characters deduction algorithm.
An advance hash table was used to provide fast and reliable access to most of the Arabic dictionary words [23]. Figure 18 shows the Arabic text spell-checking algorithm using hash table. After that, we use the minimum edit distance technique in Fig. 19 to find the suggestions of a given string. The minimum edit distance is the minimum number of operations like insertions, deletions and substitutions that is required to transform one text string into another. After the comparison processes, the words having minimum edit distance are chosen as suggestions for the incorrect string. Different algorithms for calculating distances can be used [24] [25]. The Levenshtein algorithm is the algorithm used for the weighting process. The algorithm assigns a cost of 1 for every edit operation like insertion, deletion and substitution. Those assignments are added to represent the distance between the incorrect string and the POS inside the dictionary.

```
Input: set of POS
Output: corrected set of POS.
Process:
Step 1: Look inside the dictionary using the hash table for matching. If matched, tag it as correct. If not provide a list of suggestions.
Step 2: Sort the list of suggestions and put the POS having minimal changes in the top of the list.
Step 3: If the top of the list is a one part-of-speech choose, replace it with the incorrect part-of-speech.
Step 4: If not, move to the next item inside the list and repeat the previous step.
Step 5: If reached the end of the list with no solution, use the first item in the list as the suggested correct replacement.
```

Figure 18. Arabic text spell checker algorithm.

```
Input: set of POS
Output: ordered list of suggestions
Process:
Search every part-of-speech inside the Arabic text document:
Step 1: If Arabic string was tagged as incorrect. Then, compute the minimum edit distance between the string and the dictionary POS using Levenshtein algorithm*.
Step 2: Find the minimal distance(s) between the string and the Arabic POS.
Step 3: List the suggestions in order.
*Levenshtein algorithm: for a given two strings with equal length, move one character on each string. Add 1 for every change between the two characters and 0 for a match. Add the results after reaching the end of the strings.
```

Figure 19. Arabic text spilling error detection and correction algorithm.
Figure 20 shows sample results of the Arabic POS checking algorithm.

![Sample results of the Arabic POS checking algorithm](image)

### Figure 20. Arabic POS checking results.

#### 2.2.2 Syntax Analyzer

The syntax analyzer is the second phase in the Arabic text analysis part. It has two components, Arabic POS tagging and Arabic text linking and transition. We have used a look-ahead LR (LALR) parser algorithm to build the syntax tree during the syntax analyzer phase[26][27] [28]. This LALR parser was used to handle all the grammatical and morphological “sarf” rules of the Arabic language.

**Arabic POS Tagging:**

Table 3 presents a sample of the different tagging codes for the individual POS. The result of this phase is an ordered list of POS along with different suggestions for incomplete ones. Beside each POS a tag that describes its type. Figure 21 presents the Arabic POS tagging algorithm.

<table>
<thead>
<tr>
<th>Tag</th>
<th>POS</th>
<th>Tag</th>
<th>POS</th>
</tr>
</thead>
<tbody>
<tr>
<td>VB</td>
<td>Verb, base form</td>
<td>IN</td>
<td>Preposition</td>
</tr>
<tr>
<td>VBN</td>
<td>Verb, past participle</td>
<td>JJ</td>
<td>Adjective</td>
</tr>
<tr>
<td>NN</td>
<td>Noun, singular</td>
<td>WP</td>
<td>Pronoun</td>
</tr>
<tr>
<td>NNP</td>
<td>Proper noun, singular</td>
<td>PRP</td>
<td>Personal pronoun</td>
</tr>
</tbody>
</table>
Figure 21. Arabic POS tagging algorithm.

POS Tagging Results:

Figure 22 presents the Arabic POS checking and tagging results of the algorithm in fig. 20.

```
<?xml version="1.0" encoding="utf-16"?>
<Document> ...........
  <Sentence>
    <CC text="ب" spelling="correct"/>
    <NN text="الإطار" spelling="correct"/>
    <CC text="ب" spelling="correct"/>
    <NN text="الإطار" spelling="correct"/>
    <IN text="على" spelling="correct"/>
    <NN text="الإطار" spelling="correct"/>
    <DTNN text="الأطار" spelling="correct"/>
  </Sentence>
</Document>
```

Figure 22. Arabic POS checking and tagging results.

Arabic Text Linking and Transition:

After finishing the sentences tagging, the system starts to look for linking connectivity POS inside the paragraph and whole text. There are many kinds of connectivity POS in the Arabic language. Some of those connect different sentences, other connect paragraphs, and some can do both.

Arabic Text Transition Words and Phrases:

Transition or linking words and phrases connect separate sentences and paragraphs in the same document together. All prepositions, pronouns, reference names, connected names, and enumeration words, are all considered linking tools. Furthermore, there is a couple of connectivity phrases in Arabic that also connects sentences and paragraphs. Table 4 presents some of those connectivity words and phrases.

Table 4. Some of the Arabic Connectivity Words and Phrases Between Sentences, Paragraphs and Text.
Figure 23. Arabic text linking and transition algorithm.

Each of those identifies connections are linked to one or more POS inside the paragraph or text. More than one link may be presented for a given connectivity POS. Figure 23 presents the Arabic text linking and transition algorithm. Arabic text linking and transition results are shown in fig. 24.

Figure 24. Arabic linking and transition words and phrases (Marked in red and blue).

2.2.2 Semantic Analyzer

The semantic analyzer is the third phase in the Arabic text analysis part. It has one component: the Arabic sentences checking.

Arabic Sentences Checking:

Sentences tagging starts after finishing all the POS tagging during the previews syntax analysed part. The system receives all the POS of a given sentence along with their types and characteristics. The whole sentence is analysed at this phase. The role of each POS inside the sentence is identified and the type of sentence is found[29][30][31]. We used top-bottom parsing to check and tag the sentences[32]. Figure 25 shows the Arabic sentences and phrases tagging algorithm.

Sentences Tags:

The different tags of this stage are presented in Table 5. The sentences type tagging is also presented in Table 6. **Error! Reference source not found.** presents the Arabic sentences and phrases tagging algorithm.
Table 5. POS (Sarf) Ontology Tagging

<table>
<thead>
<tr>
<th>Tag</th>
<th>Sarf Tag</th>
<th>Tag</th>
<th>Sarf Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>OEJ</td>
<td>Object</td>
<td>PRD1</td>
<td>First predicate</td>
</tr>
<tr>
<td>SUB</td>
<td>Subject</td>
<td>PRD2</td>
<td>Second predicate</td>
</tr>
<tr>
<td>CANN</td>
<td>“Can” noun</td>
<td>ENN</td>
<td>“Ena” noun</td>
</tr>
<tr>
<td>CANK</td>
<td>“Kabar Can”</td>
<td>ENK</td>
<td>“Kabar Ena”</td>
</tr>
</tbody>
</table>

Table 6. Phrase/Sentence Tagging

<table>
<thead>
<tr>
<th>Tag</th>
<th>Phrase/Sentence</th>
<th>Tag</th>
<th>Phrase/Sentence</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQ</td>
<td>Question</td>
<td>S</td>
<td>Simple clause</td>
</tr>
<tr>
<td>VP</td>
<td>Verb Phrase</td>
<td>ADJP</td>
<td>Adjective Phrase</td>
</tr>
<tr>
<td>NP</td>
<td>Noun Phrase</td>
<td>ADVP</td>
<td>Adverb Phrase</td>
</tr>
<tr>
<td>PP</td>
<td>Prepositional</td>
<td>X</td>
<td>Uncertain</td>
</tr>
</tbody>
</table>

Figure 25. Arabic sentences and phrases tagging algorithm.

Figure 26 presents a sample result of the sentences tagging results.

<?xml version="1.0" encoding="utf-16"?>
<Document> ………………………………
<S>
<CC text="و" />
<VP>
<VBD text="جعلها" />
</S>
</S>
<VP>
<VBP text="تزيد" />
<PP>
<IN text="في" />
<DTNN text="العمر" />
</PP>
</VP>
</S>
<CC text="و" />
<S>
<VP>
<VBP text="تذكر" />
<DTNN text="الأفلام" />
</VP>
</S>
</S>
</S>
</VP>
</S>
</VP>
</S>
</Document>
2.3 Intermediate Text Generator

After finishing the last phase of the text analysis part, the Arabic text is now in a tree XML format. The current subsystem is to translate the produced code of text into an intermediate text format that would be applicable for the coming subsystems to investigate and modify. The new intermediate representation is essential for the optimizer subsystem. We used the three-address code algorithm for building the rules and generating the intermediate representation. Some of the Arabic grammar rules did not obey the three-address code restriction, so we had to extend the rules to have a four-address code for some special ones. Figure 27 presents the algorithm for this intermediate text generation subsystem.

<table>
<thead>
<tr>
<th>Input: XML file</th>
<th>Output: text document with intermediate presentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process: For every sentences or phrase inside the Arabic tree XML:</td>
<td></td>
</tr>
<tr>
<td>Step 1: Look for keywords: convert the tree structure into an intermediate text specifying the rule of each string.</td>
<td></td>
</tr>
<tr>
<td>Step 2: List the intermediate representation in order.</td>
<td></td>
</tr>
<tr>
<td>Step 3: Group each paragraph intermediate representation in separate places.</td>
<td></td>
</tr>
</tbody>
</table>

Figure 27. Intermediate text generation algorithm.

Figure 28 presents a sample result of the intermediate text generation subsystem.

(a) 
\[
\begin{array}{l}
(\text{أ}) \\
\text{VP} \\
\text{VBD} /\text{أ} \text{/} \\
\text{NP} /\text{أ} \text{/} \\
\text{CC} /\text{أ} \text{/} \\
\text{NP} /\text{أ} \text{/} \\
\text{DTNN} /\text{أ} \text{/} \\
\text{PUNC} /\text{أ} \text{/} \\
\end{array}
\]

(b) 
\[
\begin{array}{l}
(\text{أ}) \\
\text{VBD} /\text{أ} \text{/} \\
\text{NP} /\text{أ} \text{/} \\
\text{CC} /\text{أ} \text{/} \\
\text{NP} /\text{أ} \text{/} \\
\text{DTNN} /\text{أ} \text{/} \\
\text{PUNC} /\text{أ} \text{/} \\
\end{array}
\]

Figure 28. Intermediate text generation sample results, (a) tree XML text format, (b) intermediate code for (a).

2.4 Modern Arabic Text Generation

2.4.1 Text Organizer

Different sentences with different POS order, and, different sentences with different synonymous for the POS are available to choose between[33][34]. Rhetorical structure theory (RST)[35][36] was used to organize the available Arabic text. RST was used to produce different forms of text using the relations between its different parts[37].
Text Organization Results:

Figure 29 presents the algorithm for the Arabic verbal sentences organizer.

![Algorithm for the Arabic verbal sentences organizer](image1)

Figure 29. Arabic verbal sentences organizer algorithm.

Figure 30 presents a sample results for the Arabic sentences organization phase.

![Sample results for the Arabic sentences organization phase](image2)

Figure 30. Arabic text organization sample results.

2.4.2 Pre-optimizer

Text summarization is one of the optimization techniques frequently used in natural language generation system [38][39]. We have used an extractive text summarization method based on Arabic conjunctive tools to summarize the ancient Arabic text. Figure 31 presents the input of the algorithm shown in fig. 32. The result of the algorithm is shown in fig. 33.

![Sentences deduction algorithm](image3)

Figure 31. Sentences deduction algorithm.
Figure 32. Arabic text summarization, sentences deduction algorithm.

Figure 33. Sentences summarization results.

Table 7 presents the results of the sentences summarization component.

Table 7. Sentences Pre-optimization Results

<table>
<thead>
<tr>
<th>Document</th>
<th>Original document (words)</th>
<th>New optimized document (words)</th>
<th>Sentences optimization</th>
<th>Sentences optimization (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>126</td>
<td>108</td>
<td>18</td>
<td>14 %</td>
</tr>
</tbody>
</table>

Figure 35 presents the input of the Arabic text summarization, words and phrases optimization algorithm shown in fig. 34. The result of the algorithm is shown in fig. 36.

Figure 34. Arabic text summarization, words and phrases optimization algorithm.

Figure 35. Words and phrases text summarization.

Figure 36. Words and phrases text pre-optimization results.

Table 8 presents the results of the words and phrases summarization part. Whereas, Table 9 presents the overall results of the pre-optimization phase.
Table 8. Primary Words and Phrases Summarization Results.

<table>
<thead>
<tr>
<th>Document</th>
<th>Sentences Optimized document (words)</th>
<th>New optimized document (words)</th>
<th>Words and phrases optimization</th>
<th>Words and phrases optimization (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of words</td>
<td>108</td>
<td>91</td>
<td>17</td>
<td>16 %</td>
</tr>
</tbody>
</table>

Table 9. Overall Primary Summarization Results.

<table>
<thead>
<tr>
<th>Document</th>
<th>Original document (words)</th>
<th>New optimized document (words)</th>
<th>Words and phrases optimization</th>
<th>Words and phrases optimization (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of words</td>
<td>126</td>
<td>91</td>
<td>35</td>
<td>28 %</td>
</tr>
</tbody>
</table>

2.4.3 Semantics Generator

Arabic POS Synonymous Generator:

Synonymy is one of the lexical semantic relations (LSRs). It represents the relation between the meanings of the different words. [40]. Using the dictionary of Arabic to Arabic words “Ma’jam”, the synonymous of all the words inside the ancient text document are obtained. A special manually built “Ma’jam” was prepared for this purpose. Arabic words with their different synonymous were added to this list along with other features. An important feature is the subject of the synonym. This field represents the domain of usage. Each feature in the dictionary has a code representing its different values. The available synonymous and their features together provide the most suitable alternative for the given word[41]. Synonymous for a given word inside a sentence are placed in the original word place provided that both have the same POS type. Sentences that do not have synonymous for any of its POS would be left without any modifications. Figure 37 shows the Arabic synonymous generator algorithm, and fig. 38 shows a sample results of the algorithm. Figure 39 shows the algorithms sample results after removing confusions.

![Arabic synonymous generator algorithm](image)

**Input:** text document  
**Output:** text document with words replaced by their synonymous

**Process:**  
For every word or phrase inside the Arabic text document:  
*Step 1:* Look inside the handmade list of Arabic synonymous (Ma’jam), if found a synonymous, replace the original word or phrase with the proper synonym.  
*Step 2:* After reaching the end of document go to the next step.  
*Step 3:* Starting from the beginning of the document remove any confusions on sentence like the existence of two matching words inside the same sentence, or unneeded conjunction.

Figure 37. Arabic synonymous generator algorithm.
Table 10 presents the results of the synonymous generator component.

<table>
<thead>
<tr>
<th>Document</th>
<th>Original Document</th>
<th>Synonymous found</th>
<th>Synonymous found (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of words</td>
<td>91</td>
<td>22</td>
<td>24 %</td>
</tr>
</tbody>
</table>

### 2.4.4 Post-optimizer

The post optimizer phase uses the same optimization algorithms used in the pre-optimizer phase shown in fig. 32 and fig. 34. The two algorithms are applied to the output text generated after the semantics generator phase. Figure 40 presents the input text for this phase. Potential sentences to be removed by the first algorithm are underlined. Figure 41 shows the results after applying the Arabic text summarization and sentences deduction algorithm.

<table>
<thead>
<tr>
<th>Document</th>
<th>Original document (words)</th>
<th>New optimized document (words)</th>
<th>Sentences optimization</th>
<th>Sentences optimization (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of words</td>
<td>115</td>
<td>107</td>
<td>8</td>
<td>7 %</td>
</tr>
</tbody>
</table>

Table 11 presents the results of the sentences summarization component.
Figure 42 shows the results after applying the Arabic text summarization and words and phrases deduction algorithm.

Table 12 presents the results of the words and phrases post-optimization component. Whereas, Table 13 presents the overall results of the optimization phase.

Table 12. Words and Phrases Summarization Results.

<table>
<thead>
<tr>
<th>Document</th>
<th>Sentences Optimized document (words)</th>
<th>New optimized document (words)</th>
<th>Words and phrases optimization</th>
<th>Words and phrases optimization (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of words</td>
<td>107</td>
<td>105</td>
<td>2</td>
<td>2%</td>
</tr>
</tbody>
</table>

Table 13. Overall Summarization Results.

<table>
<thead>
<tr>
<th>Document</th>
<th>Original document (words)</th>
<th>New optimized document (words)</th>
<th>Words and phrases optimization</th>
<th>Words and phrases optimization (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of words</td>
<td>126</td>
<td>105</td>
<td>21</td>
<td>17%</td>
</tr>
</tbody>
</table>

3. CONCLUSIONS

A novel approach to recognizing text in Arabic ancient manuscripts was fully implemented and tested using different ancient Arabic manuscripts. The first phase of the image recognition part, the binarization phase, produced better results using the new hybrid binarization algorithm than two of the well-known binarization algorithms. As expected, the training process of the system took a huge amount of processing time especially when the training set was large. After that, the recognition process was considerably fast and the recognition rate reached 71 percent.

The text organizer phase produced accurate results for the available sentences organization algorithms. The output of this subsystem depends on the added rules; more rules generate more different forms of a sentence but on the expense of the system speed. The pre-optimizer phase is performing the optimization with 28 percent. The semantics generator found synonymous for 24 percent of the text in the test manuscript. This phase was fast and greatly depends on the size of the synonymous list. The post-optimizer phase is done on the text produced by the semantics generator phase. The overall optimization of the ancient text after the post-optimizer phase reached 17 percent of the total word count.

Our future work will focus on the following points:
- Improve the current system by improving the existing segmentation techniques for PAWs using individual parts-of-lines with similar features. And, add new features to the classifier to improve recognition rate.
- Expand the system to deal with the ancient Arabic manuscripts with handwritten styles that had no dots or diacritics by adding a new smart part to the system to add those dots and diacritics.
- Build a database or set for ancient Arabic manuscripts to improve the research on binarization and recognition.
- Build an Arabic dictionary system along with a tagging system for the grammatical and morphology “sarf” rules.

REFERENCES

International Journal on Natural Language Computing (IJNLC) Vol. 4, No.6, December 2015


