The International Young Researchers' Conference
DOI: doi.org/10.34614/iyrc0038

How does a computer understand Modern Standard Arabic's morphological scales?

Saughmon Boujkian
UWC Dilijan, Armenia
s.boujkian@student.uwcdilijan.am

Abstract
Due to my interest in Linguistics, I've always been passionate about studying the morphological scales in Modern Standard Arabic. Later, I found out about their importance in finding the plural forms of the nouns and realized a lack of a program that identifies them. Identifying them using a computer will help Modern Standard Arabic learners to understand them better, and make sure the scales they follow are right and usable. This drove me to start researching different ways in which I could program software that finds these morphological terms. As a result, I started finding patterns in the morphological terms, focusing mainly on the place in which the root-letters are found. Afterward, I found a way to program them and finally tested said program. Finally, I could program software that is able to identify not only the morphological scales but also whether these scales exist or not.
Keywords: morphological scales, Modern Standard Arabic, automated translation

Morphological Scales in Modern Standard Arabic

Morphological scales are a way in which words can change from verbs to nouns to give the meaning of either the subject, the object, or the instrument to which the verb was attributed to. In Modern Standard Arabic, morphological scales are used to give more accurate meanings to the words, and to group nouns. In MSA, there are some words that do not have a straightforward plural form, and the easiest way to identify them, apart from the native speakers' intuition, is to find their morphological scale. Since most nouns with the same morphological scales have the same method in which the plural form is done.

In Figure 1, you can find an example that explains morphological scales in modern standard Arabic.

As seen in Table 1, the verb is usually a form of three letters called the root-letters. Special cases occur, where a root might have up to five letters. These cases will be ignored in this work for simplicity, as they are uncommon and can be handled in an analogous manner to the ones presented in this paper. These three root-letters are combined with both other letters and diacritical marks in order to form one of the scales. For example, a scale that gives the meaning of the do-er of the verb, or the meaning of the subject scale. The scale that gives the meaning of the do-er of the verb is done by adding an ''a'' after the first letter, and an e diacritical mark on the second root letter. So ''ktb'' would become ''kateb''.

FIGURE 1. Letter numbering in a word.

Mathematical Representation of the Morphological Scales

As observed in Figure 1, the positions of the root-letters change according to the scale. For example, if the scale is the subject one, the root letters will be the first, the third, and the fifth letters of the word. This pattern is the same for all other verbs, such as "fth", meaning "to open", where "fateh" is the subject scale form. As such, a static code of three numbers can be assigned to represent the subject scale, this code is 135. Each of the morphological scales has different positions of root-letters, and hence, a fixed code is assigned to them.

TABLE 1. Explanation of morphological scales.

To write	ktb كَتَبَ
Writer, so it is called the subject scale. *(Abaiat, 2017)*	Kateb كَاتِب
The thing that's written, so it is called the object scale. *(Alfadelat, 2020)*	mktub مكتوب
The one who writes a lot, so it is called the exaggerated object scale. (*Nafham, 2016)*	kttab كَتَّاب

These codes differ when considering Arabic script, since diacritical marks, as used in the subject scale, for example, are not considered a letter in the Arabic language. This means that "kateb" would be "katb" with an "e" diacritical mark on the "t".

Amid researching, I found out that it is possible to use Arabic characters as input. Thereby, I have updated the list of static-codes in correspondence to the positions of the letters when written in Arabic. Table 2 shows the codes as used in my program;

TABLE 2. Special codes for the scale.

Scale	Static codes
Subject scale	023
Object scale	124
Subject-like adjective scale	012, 013,
Exaggerated subject scale	013, 120, 124, 013, 012

I have committed to zero-based indexing in my codes to maintain consistency with the programming language in use, i.e. Python. This means that the first letter of a word has a position zero instead of one. This explains the codes generated in the table above.

In the table above, some codes appear multiple times. In these cases, I cannot depend merely on the given code to identify the morphological scale. Hence, I examine the words in more details as follows:

For the code "013", I distinguish between a subject-like adjective scale and an exaggerated subject scale by the following: (Atiah, 2019)

- If the letter at position 2 is ''e'' or "u", then it is an exaggerated subject scale.
- If the letter at position 2 is ''a'', then it is a subject-like adjective scale.

Therefore, by merging the conditions above with the codes, the scales can be identified.

Some root-letters do not stay the same when converting the base-form into another scale. Below, I list the different cases where this phenomenon happens, along with the used mechanism to identify them:

1. If the second root-letter is an “a” like “kal”, the letter is converted to ئ in the subject scale. (Abaiat, 2017)
2. If the third root-letter is an ا , it is converted to و in most of the scales. (Atiah, 2019)
3. If the third root-letter is an ى, it is converted to ي in most of the scales. (Atiah, 2019)
4. If the first root-letter is an أ, it is converted to an آ in the subject scale. (Abaiat, 2017)

Each one of these complex cases will be analyzed, and a way to examine them on a computer will be discussed in the next section.

Programming the Rules and the Static Codes

My decision to use Python as the programming language was based on the fact that Python is one of the most widespread languages for linguistic application. It is also due to my previous experience in this language.

In order to start programming, first, the root-letters should be identified, therefore, input from the user is needed.

As shown in Figure 2a, The user is asked to enter the first, the second, and the third root letters. Each of these root-letters is stored in a different variable in order to be used later. After that, the user is asked to enter the word.

As shown in Figure 2b, the word is also saved into a variable in order to be processed later.

Going back to the root-letter changes, ى changes into ي, this is reflected by the ‘’if ‘’statement as shown in Figure 3a.

Moreover, as previously mentioned, if the second root-letter is an ‘’a’’ it is changed to either ‘’i’’ or ‘’u’’, therefore additional input from the user is required in order to check whether it is changed to an ‘’i’’ or an ‘’u’’.

```
first_root = input("  :أدخل فاء الفعل")
second_root = input("  :أدخل عين الفعل")
third_root = input("  :أدخل لام الفعل")
```

Figure 2 a

```
word = input(" :أدخل الكلمة ")
```

Figure 2 b

FIGURE 2. (a) Root-letter input. (b) Word input.

After gathering all the necessary information about the word, here is the method I used to obtain the three-digit code;

As shown in Figure 3b, the ‘’if’’ statement excludes some special cases that could be extremely hard to handle using merely letter-positions. Using the index function, the position of each root letter in the word is obtained. For example, if the first root letter is the first letter in the word, the index would give 0, etc. I concatenate the computed positions at the end to get the final code.

In the following, I describe how by using these three-digit codes, each scale can be identified.

```
if third_root == "ى":
    third_root = "ي"
```

Figure 3 a

```
if third_root != "ا" and third_root != "ي" and second_root != "ا" and first_root != "ا" and second_root != "ا":
    one_no = str(word.index(first_root))
    three_no= str(word.index(third_root))
    two_no = str(word.index(second_root))
    sum = (one_no) + (two_no) + (three_no)
```

Figure 3 b

FIGURE 3. (a) A special case. (b) Code creation.

Subject Scale

As shown in Figure 4a, for the subject scale, if the code, which is stored in the sum variable is “023”, and there is no ‘’a’’ in the second root letter, -in other words, the verb is not an exception- the program will tell the user that it is a subject scale (اسم فاعل).

As seen in Figure 4b, if the verb is one of the special cases, another if statement is required. If the first letter of the word is آ, and the second letter is the same as the second root-letter, it means that it is a subject scale word.

```
if sum == "023" and second_root != "ا":
    print("اسم فاعل")
    type == "اسم فاعل"
    exit()
```

Figure 4 a

```
if word[0] == "آ" and word[1] == second_root:
    print("اسم فاعل")
    type == "اسم فاعل"
    exit()
```

Figure 4 b

FIGURE 4. (a) Subject scale without special cases. (b) Subject scale with special cases.

Object Scale

In this case, static codes can be confusing and are not necessarily needed. Deciding object scale is hence done using the following if statement.

We see in Figure 5a that, if the first letter of the word is “m”, -since object scale always starts with an “m”- the second word of the letter is the same as the first root-letter. The fourth letter of the word is not an a, a special case- and the second letter of the word is not ؤ, - another special case- it means that this is an object-scale word.

In order to take the special cases into consideration, I added the following;

If the second root-letter is an ‘’a’’, additional input is required, whether it changes into a ‘’u’’ or an ‘’i’’, and saved into a variable called ‘’asel’’. The variable that contains the third root-letter is changed to “asel”, and the same process of finding the code is repeated again. In this case, the special case will be identified and the scale will be given.

```
if word[0] == "م" and word[1] == f and word[3] != "ا" or word[1] == "ؤ":
    print("اسم مفعول")
    type == "اسم مفعول"
    exit()
```

Figure 5 a

```
if sum == "124":
    print("مبالغة اسم فاعل على وزن مفعال")
    type == "مبالغة اسم فاعل على وزن مفعال"
    exit()
```

Figure 5 b

FIGURE 5. (a) Object scale without special cases. (b) Special code for exaggerated subject scale.

Exaggerated Subject Scales

As shown in Figure 5b, one of the scales for the Exaggerated subject has its unique code, which is “124”. Therefore if the code is “124”, this is an exaggerated scale word.

And as shown in Figure 6a, if the sum is “013”, and the second letter of the word is a ‘’u’’ or an ‘’i’’, it means this is a word on the exaggerated subject scale.

```
if sum == "013" and word[2] == "و":
    print("مبالغة اسم فاعل على وزن فعول")
    type == "مبالغة اسم فاعل على وزن فعول"
    exit()
```

Figure 6 a

```
if sum == "012" and word[3] != "ا" and word[4] != "ن":
    if harakesum == "ؤ":
        print("مبالغة اسم فاعل على وزن فُعل")
        type == "مبالغة اسم فاعل على وزن فُعل"
    exit()
```

Figure 6 b

FIGURE 6. (a) Another case for an exaggerated subject scale. (b) Different case for an exaggerated subject scale.

The Subject-like Adjective Scales

Here, some of the subject-like adjective scales are exactly the three root-letters together but with different diacritical marks, that's why if the word is just the same as the three root letters together, the user is asked to input the first two letters' diacritical marks. The combination of these first two letters' diacritical marks will help to identify some of the subject-like adjective scales.

In Figure 6b, we see that if the code is "012", there are no special cases, and the combination of the diacritical marks is only an ''u'', it means that this is a subject-like scale. Some other cases were presented as well. Moreover, some of the subject-like adjective scales have a different female version, and these were taken into consideration as follows;

Case Studies

I present three case studies with corresponding analyses to make sure the program functions correctly.

As shown in Figure 7a, the program asked for the three root letters and then a word. It was a simple word without any special cases, and the program could identify it as the subject scale. It is correct since the word means the writer, which is obviously the doer of the verb, therefore the subject.

In Figure 7b, the word contains an ''a'' as its second root-letter, which means that it is a special case. The program identified it successfully since it mentioned that it is an object scale, and the word means something that is said, meaning it is the object.

In Figure 7c, the word is the same as the three root-letters but with different diacritical marks, after identifying the diacritical marks, the computer failed to identify the scale. This is because the word does not exist and the diacritical mark order is wrong.

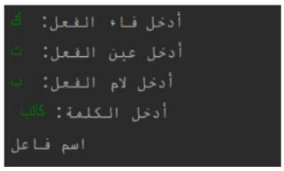

Figure 7 a

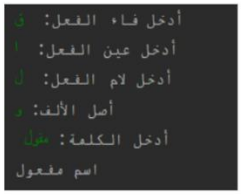

Figure 7 b

Figure 7 c

FIGURE 7. (a) Case study one. (b) Case study two. (c) Case study three.

Conclusion and Evaluation

As presented, the computer could identify the scales when they actually exist and fail to when a word is made up. This means that the computer could not only identify to which scale the word belongs, but also whether it exists or not. This identification is extremely important in the development of the technical use of Modern Standard Arabic, seeing as these scales are used in order to get the plural of the word. If the computer knows which scale the word is, it will immediately know how to give the plural of the word, therefore being able to translate texts into Arabic faster and more accurately.

One possible shortage of my work is that it only makes use of if statements and does not use more advanced functions. Even though this keeps the program simple and comprehensible, using more advanced and language-specific functions could make the program shorter. One of these functions is the ''Class function''. All the scales have common roots, which means that they can be identified depending on specific aspects. Using functions, I could systematically look for specific characteristics in words and identify them, which can reflect the word's scale. To go even further, the class function could be used in order to develop the program.

Some of the benefits of this research are that it gives the feminine version for words whose feminine version is irregular. These feminine

versions of the words can be confusing to some learners of MSA, and if the program gave it to them, it will help them to learn it better and use its right form. Moreover, another strength point is its diversity in identifying irregular cases and not only the regular ones. Again, the irregular cases can be extremely confusing for learners, and having such a program will help them to check their understanding.

This program is a start to a bigger project to help translation into Arabic get more automated. By identifying the morphological terms, translating plural nouns will be easier since its MSA morphological term is already identified, and as mentioned in the introduction, morphological terms give the plural form of the nouns. The next steps will be learning about how to make program data processing, then applying it to make a database of words and their morphological terms. This would make it more accessible to anyone who needs it and ease the development of the next step, which would be improving the automated translation of MSA.

References

Abaiat, L. (2017). The Definition of the Subject Scale. Retrieved from shorturl.at/tEZ09.

Alfadelta, H. (2020). The Definition of the Object Scale. Retrieved from shorturl.at/xFR59

Atiah, A. (2019). The Scales of the exaggerated subject, and the adjective-like subject. Retrieved from: shorturl.at/rDS38

Nafham. (2016). Subject scale, and exaggerated subject scale. Retrieved from https://www.youtube.com/watch?v=EXBYFFERGh0