Academic Journal of Research and Scientific Publishing | Vol 3 | Issue 32
Publication Date: 5-12-2021 ISSN: 2706-6495

A Template-Based Approach for Tagging Non-Vocalized Arabic Nouns

By: Mr. Hashem Saadaldin Alghalib Alsharif

Managing Director Deanship of Admission & Registration King Abdulaziz University
Jeddah, Kingdom of Saudi Arabia

Email: halshareef@kau.edu.sa

Abstract:

There exist no corpora of Arabic nouns. Furthermore, in any Arabic text, nouns can be found in different forms. In fact, by tagging nouns in an Arabic text, the beginning of each sentence can determine whether it starts with a noun or a verb.

Part of Speech Tagging (**POS**) is the task of labeling each word in a sentence with its appropriate category, which is called a **Tag** (*Noun, Verb and Article*). In this thesis, we attempt to tag non-vocalized Arabic text. The proposed POS Tagger for Arabic Text is based on searching for each word of the text in our lists of Verbs and Articles. Nouns are found by eliminating Verbs and Articles. Our hypothesis states that, if the word in the text is not found in our lists, then it is a Noun. These comparisons will be made for each of the words in the text until all of them have been tagged.

To apply our method, we have prepared a list of articles and verbs in the Arabic language with a total of 112 million verbs and articles combined, which are used in our comparisons to prove our hypothesis.

To evaluate our proposed method, we used pre-tagged words from "*The Quranic Arabic Corpus*", making a total of 78,245 words, with our method, the Template-based tagging approach compared with (AraMorph) a rule-based tagging approach and the Stanford Log-linear Part-Of-Speech Tagger.
Finally, AraMorph produced 40% correctly-tagged words and Stanford Log-linear Part-Of-Speech Tagger produced 68% correctly-tagged words, while our method produced 68,501 correctly-tagged words (88%).

Keywords: Template, Approach, Tagging, Non-Vocalized, Arabic Nouns.

1. Introduction

Arabic texts can be either: **Vocalized** (*The Holy Quran*) or **Non-Vocalized** (*newspapers, books, and the media*). Handling non-vocalized texts is confusing; due to the ambiguity problem, words may have more than one meaning. "كتب" (*this can be a noun "books" or a verb "to write"*).
There exist no corpora of Arabic nouns. Furthermore, in any Arabic text, nouns can be found in different forms and can refer to non-Arabic items or things. In fact, by tagging nouns in an Arabic text, the beginning of each sentence can determine whether it starts with a noun or a verb.
In this chapter, we will discuss Word Templates, Natural Language Processing, and Part of Speech Tagging (**POS**), then we will state out objectives and the hypothesis used in this thesis, followed by an introduction to the Arabic words lists used in our approach. Finally, the structure of our thesis will be outlined.

1.1 What is a Word Template?

The term 'Word Template' is defined as any processing element that can be combined with a data model and processed by a template engine to produce a result (Ndie et al., 2010).

1.2 Natural Language Processing

Any language that is naturally used by humans (*Arabic and English*) is called Natural Language. In other words, a natural language is not an artificial or a man-made language, such as programming languages. Therefore, Natural language processing (NLP) is an area of research and application that explores how computers can be used to understand and manipulate natural language text or speech to do useful things(Chowdhury, 2003).

1.3 Part of Speech Tagging

Part of Speech Tagging (**POS**) is the task of labeling each word in a sentence with its appropriate category, which is called a **Tag** (*Noun, Verb and Article*) (AlGahtani et al., 2009). In this thesis, we attempt to tag non-vocalized Arabic text.

We intended to use a simplified tags set in our approach (*Noun, Verb and Article*) because our objective is to tag non-vocalized Arabic text, by applying a Template-based approach without explaining the construction of the Arabic sentence and also because most of the Arabic language is rooted to a verb, noun or article.
POS tagging is useful for a large number of applications. It is the first analysis step in many syntactic parsers; it is required for the correct lemmatization of words; and it is used in information extraction, vocalization, translation, speech synthesis, lexicographic research, term extraction, and many other applications.

1.4 Objectives

There are many methods of POS tagging which can be classified into three categories: the Statistical Approach, the Rule-Based Approach and the Hybrid Approach. All of these approaches are CPU consuming while our approach is memory consuming. Since the 1980’s, processor speed has increased by 1,000 times. Meanwhile, memory capacity has increased by 1,000,000 times.

It has been found that, since 2005, CPU speed has not followed the estimation of Moore's law. In fact, due to the thermal wall – *as he described it* – CPU speed has reached a certain limit(Benson, 2014).

In Figure 1, you can see that transistor density has continued to climb according to Moore's Law. However, power consumption and CPU have reached a certain limit.

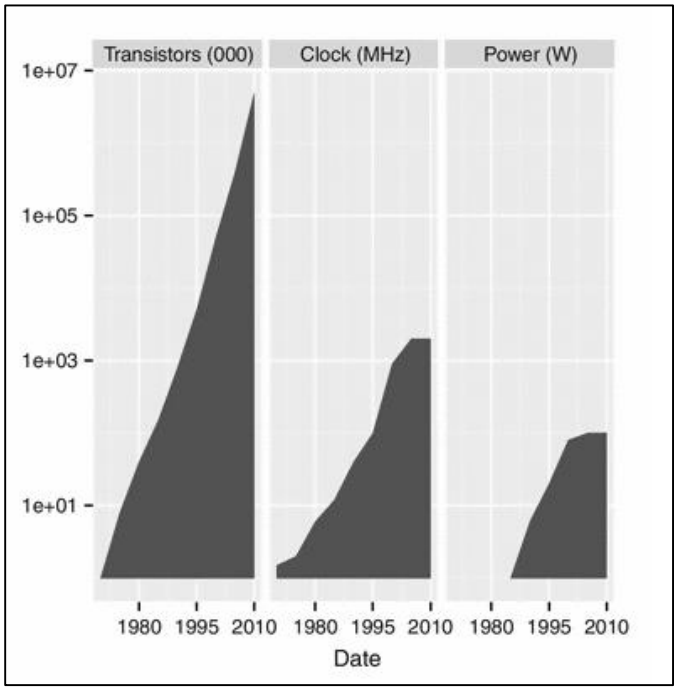

Figure 1: Transistor, clock, and power in Intel CPUs over time.

Meanwhile, the physical memory capacity is still growing; for example, Windows 7 64-bit memory limit is up to 192 GB.

Version	Limit on X86	Limit on X64
Windows 7 Ultimate	4 GB	192 GB
Windows 7 Enterprise	4 GB	192 GB
Windows 7 Professional	4 GB	192 GB

Figure 2: This table specifies the limits on physical memory for Windows 7

For the above, using a memory consuming approach is more appropriate than a CPU consuming one.

1.5 Our Hypothesis

The proposed POS Tagger for Arabic Text is based on searching for each word of the text in our lists of Verbs and Articles. Nouns are found by eliminating Verbs and Articles.

For example, a simple sentence like (حضر زيد إلى المدرسة) can be tagged by comparing the first word (حضر) with our Articles list. The word will be tagged "*Article*" if it is found. Otherwise, the same word will be compared with the Verb list.

The word will be tagged "*Verb*" if it was found. Otherwise, by default, the word will be tagged "*Noun*". Our hypothesis states that, if the word in the text is not found in our lists, then it is a Noun. These comparisons will be made for each of the words in the text until all of them have been tagged.

This approach can be applied to Arabic language because it has static verb derivations and a limited number of words.

1.6 Arabic Verbs and Articles Lists

To apply our method, we must first prepare a list of articles and verbs in the Arabic language. Unfortunately, we only have a list of three-lettered verbs (*over 28 million verbs*) produced as text files by the Arabic Language Template Grammars Component Based Technology (Ba-Aziz, 2009).

Four-lettered verbs (*about 2,000 verbs*) and articles (*over 4,000 articles*) of the Arabic language have not been employed before. All of our data, both new and old, must be prepared for our method, and we must add the proper prefix to the whole data.

The result was a total of 112 million verbs and articles combined, which are used in our comparisons to prove our hypothesis.

As will be discussed in sections 2.2.4 and 2.2.5, using a program that we have developed, a list of all four-lettered verbs and articles will be produced.

1.7 Thesis Layout

The next chapter, Chapter Two, "*Literature Review*," will provide a brief yet better understanding of the Arabic language, together with its rich and complex morphology, and also survey some of the similar work previously performed on the application of POS tagging approaches to Arabic.

In Chapter Three, "*Methodology*," we will discuss the methodology we applied to prove our thesis, and also provide the necessary methodology to design the system as well as analyze its different categories. The implementation of our model will be presented along with the evaluation method that will be used to test the proposed model fully.

Finally, in Chapter Four, "*Results and Conclusions*," a full test of the proposed model will be conducted and a comparison with the two other systems will be undertaken. Finally, a conclusion of the results at hand will be presented.

2. Literature Review

2.1 Introduction

POS tagging is one of the most important natural language problems studied by researchers. The significance of POS tagging for language processing is the large amount of information it provides about a word and its neighbors (Jurafsky & Martin, 2000).

POS tagging is the process of assigning a part-of-speech or other syntactic class marker to each word in the corpus (Jurafsky & Martin, 2000). It is, in other words, the process of assigning a tag from a limited set of tags (*Tag Set*) to a word. The number of tags in a tag set depends on both the language and the intended application. If we talk about tagging, then we always mean some tag set, perhaps implicitly.

There are many methods associated with POS tagging. Most of the modern methods use some form of machine learning.

In this chapter, we will present an introduction to Arabic Language and Arabic Morphology, then outline the POS tagging approaches used for Arabic.

2.2 The Arabic Language

Arabic is ranked sixth in the world's league table of languages. With 22 countries defining Arabic as their official language, it has an estimated 186 million native speakers and is spoken by at least 250 million people. As the language of the Quran, the holy book of Islam, it is also widely used throughout the Muslim world. The Arabic language belongs to the Semitic group of languages, which also includes Hebrew and Amharic, the main language of Ethiopia.

Arabic is one of the few languages that exhibits diglossia, which is the separation between the spoken language (*dialects*) and the formal language. However, the formal language is Modern Standard Arabic (**MSA**), which is used in written texts and spoken in formal settings (Zughoul & Abu-Alshaar, 2005).

Arabic is written using the Arabic alphabet, from right to left, and there are no capital letters. Arabic has two genders: masculine and feminine.

Arabic letters are linked together to form words, so some letters may change shape according to their location in the word.

There are 29 letters in the Arabic language, with seven letter sounds that do not exist in the English language (غ،ع،ط،ض،ص،خ،ح). Arabic uses vocalized symbols to determine the sound of a word, and these symbols may or may not be written down, resulting in a very high level of ambiguity.

In addition to the singular and plural constructs, Arabic has an added form called the "*dual*", which indicates precisely two of something. Arabic has only three verb tenses: the past, the present, and the imperative.

2.2.1 Arabic Morphology

Morphology is concerned with the structure of words. It is almost inconceivable for a natural language application not to employ morphological knowledge (Wintner, 2004).

Like any other Semitic language, Arabic is highly inflected. Based on what is normally called a "*Constant Root System*", words are derived from a root and pattern, combined with prefixes, suffixes, and circumfixes.

The root, which contains the seed meaning of the word, consists of 3-4 constants, that are called radicals, and the pattern or a "*template*" is a sequence of variables. Patterns are simply all of the different variations of prefixes, infixes, and suffixes that can be allocated to any given root.

Arabic compound words are created by assigning the root radicals to the pattern variables. The combinations of the same root with a different pattern may result in a different meaning.

Sometimes, prefixes and/or suffixes are attached to words. Those affixes may modify several features of the word, including its number (*singular, dual, plural or collective*), gender (*masculine, feminine or no gender*), possession, definiteness, case (*nominative, accusative, or genitive*), tense (*past, present, and future*) and more.

Arabic affixes have the feature of concatenating with each other according to predefined linguistic rules, which increases the overall number of affixes (Al-Sughaiyer & Al-Kharashi, 2004).

2.2.2 Arabic Vowel Marks

In Arabic, there are three kinds of vowels:

- The three vowel letters, which are (ا), (و) and (ي). These are used for long vowels.
- The "Hamza".
- The vowel marks which are used to denote short vowels.

To distinguish short vowels from long ones when words are read, Arabic script uses vowel marks. This is implemented by writing the marks over or under a letter.

In modern day Arabic text, these marks are usually not written because Arabic readers can always guess them. Therefore, in our approach, we will consider a tagging system for non-vocalized Arabic text (*a text with no vowel marks*).

2.2.3 Arabic as a Templatic Language

Perhaps the most interesting aspect of Arabic is its regularity of form. Arabic morphemes "*words*" generally derive from three root radicals to which various affixes (*prefixes, suffixes, and infixes*) can be attached to create a word (Seikaly, 2207).

2.2.4 Four-Lettered Verb List

As will be described later in Figure 3, a verb is divided into two main types: complete (تام), as in (عسكر), and deficient (معتل), as in (وسوس).

A complete verb, as in (عسكر), is again divided into two types: intransitive (لازم), as in (بعثر), and transitive (متعد), as in (تزلزل).

A transitive verb, as in (تزلزل), is divided into two types: passive (مبني للمجهول) and active (مبني للمعلوم).

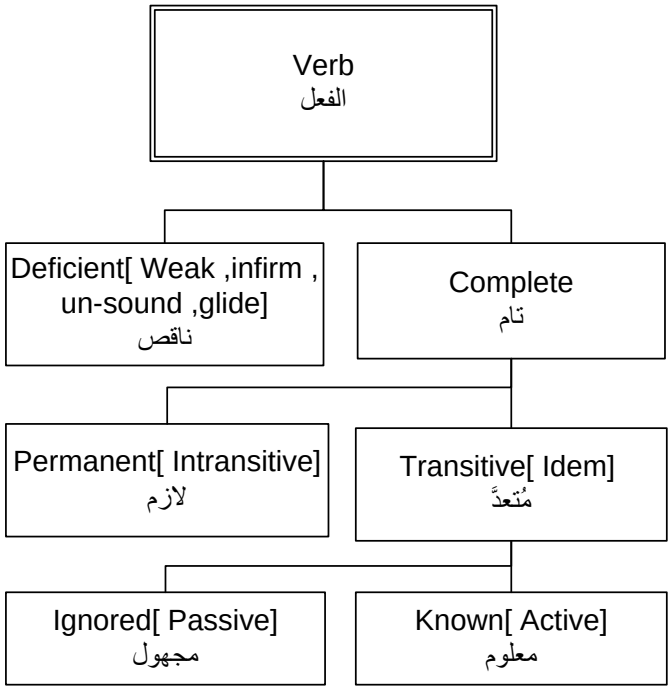

Figure 3: Arabic Verbs

All these types must be considered in order to create a complete list of four-lettered Arabic verbs. Figure 4 describes the morphology rules used to generate all Arabic verbs.

		Number	Personal Pronoun		
			غائب 3rd	مخاطب 2nd	متكلم 1st
Masculine	مذكر Male	Singular مفرد	هو	أنت	أنا
		Dualمثنى	هما	أنتما	نحن
		Plural جمع	هم	أنتم	نحن
	مؤنث Feminine	Singular مفرد	هي	أنتِ	أنا
		Dual مثنى	هما	أنتما	نحن
		Plural جمع	هن	أنتن	نحن

Figure 4: Morphology rule attributes for Arabic verbs

For example, or the four-lettered Arabic verb derivation, we considered the verb (عسكر). As shown in Figure 5, a complete derivation of the verb (عسكر) was presented.

	الماضي	المضارع	المضارع المجزوم	المضارع المنصوب	المضارع المؤكد	الأمر	الأمر المؤكد
أنت	عسكرت	تعسكر	تعسكر	تعسكر	تعسكرن	عسكر	عسكرن
أنت	عسكرت	تعسكرين	تعسكري	تعسكري	تعسكرن	عسكري	عسكرن
أنتما	عسكرتما	تعسكران	تعسكرا	تعسكرا	تعسكران	عسكرا	عسكران
أنتم	عسكرتم	تعسكرون	تعسكروا	تعسكروا	تعسكرن	عسكروا	عسكرن
أنتن	عسكرتن	تعسكرن	تعسكرن	تعسكرن	تعسكرنان	عسكرن	عسكرنان
أنا	عسكرت	أعسكر	أعسكر	أعسكر	أعسكرن		
نحن	عسكرنا	نعسكر	نعسكر	نعسكر	نعسكرن		
هو	عسكر	يعسكر	يعسكر	يعسكر	يعسكرن		
هي	عسكرت	تعسكر	تعسكر	تعسكر	تعسكرن		
هما	عسكرا	يعسكران	يعسكرا	يعسكرا	يعسكران		
هم	عسكروا	يعسكرون	يعسكروا	يعسكروا	يعسكرن		
هن	عسكرن	يعسكرن	يعسكرن	يعسكرن	يعسكرنان		

Figure 5: An example of a four-lettered Arabic verb derivation.

In our approach, which is based on comparisons, all of the verbs and articles in the Arabic text must be denoted in our lists in order to define Nouns by elimination. Therefore, it is necessary to add the proper prefix and suffix. Many of the prefixes and suffixes are generated within derivations of verbs and articles. However, we have added the missing prefixes as shown in the figure below.

```
for line in infile:
    if len(line)>2:
        outfile.write(line + u'\r\n')
        outfile.write(u"و" + line + u'\r\n')
        outfile.write(u"ف" + line + u'\r\n')
        outfile.write(u"ل" + line + u'\r\n')
        outfile.write(u"س" + line + u'\r\n')
    else:
        outfile.write(line)
```

Figure 6: The function used to add prefixes to Verbs and Articles.

Naturally, since we are focusing on tagging non-vocalized Arabic text, many derivations of the same verb will be duplicated.

Therefore, we will have a redundancy problem. In Figure 5, after extensive revision, it is clear that most of the verb (عسكر) derivations are the same. Deleting these duplicates is mandatory.

```
def del_dups(seq):
    seen = {}
    pos = 0
    for item in seq:
        if item not in seen:
            seen[item] = True
            seq[pos] = item
            pos += 1
    del seq[pos:]
```

Figure 7: The function employed to delete duplication.

Finally, by using a program that we developed, we found and listed the total of over 2,000 four-lettered Arabic verbs, with all of their unique derivations.

2.2.5 Articles List

In Arabic language, there are about 80 articles, known as: (حروف المعاني) (Wright & Caspari, 2011).

الضمير	أنت	أنتما	أنتم	أنتِ	أنتن	هو	هما	هم	هي	هن	أنا	نحن
آي	آيك	آيكما	آيكم	آيك	آيكن	آيه	آيهما	آيهم	آيها	آيهن	آيي	آينا
إلي	إليك	إليكما	إليكم	إليك	إليكن	إليه	إليهما	إليهم	إليها	إليهن	إلي	إلينا
أن	أنك	أنكما	أنكم	أنك	أنكن	أنه	أنهما	أنهم	أنها	أنهن	أني	أننا
إن	إنك	إنكما	إنكم	إنك	إنكن	إنه	إنهما	إنهم	إنها	إنهن	إني	إننا
أي	أيك	أيكما	أيكم	أيك	أيكن	أيه	أيهما	أيهم	أيها	أيهن	أيي	أينا

Figure 8: An example of Arabic articles derivation.

Applying the morphology rules used to generate all Arabic verbs to all Arabic articles using a program that we have developed, keeping in mind the redundancy problem, we managed to list a total of over 3,000 articles with all of their unique derivations.

2.3 POS Tagging Approaches used for Arabic

2.3.1 SVM

Diab (Diab et al., 2004), (Diab, 2009) applied a Support Vector Machine (**SVM**) to Arabic POS tagging and tokenization.

The SVM-POS tagger achieved an accuracy level of 95.49%. The Arabic TreeBank, consisting of 4,519 sentences, was used for the training and testing. It employs the LDC's POS tagset, which consists of 24 tags35.

2.3.2 SVM + Morphological Analyzer

In Habash (Habash & Rambow, 2005), a Support Vector Machine (**SVM**) was applied with the support of a morphological analyzer to produce all possible analyses. Their POS evaluation shows an accuracy of 97.6% on ATB1 and an accuracy of 95.7% on ATB2, both of which are based on gold standard tokenization.

2.3.3 Statistical and Rule-based

In Khoja (Khoja, 2001), a system is developed, using a combination of both statistical and rule-based techniques. It uses a simple tagset.

A corpus of 50,000 Modern Standard Arabic words (*an extract from the Saudi Al-Jazirah newspaper, dated 03/03/1999*) was tagged using this tagset 36. It achieved an accuracy of around 90%.

2.3.4 Stanford PoS Tagger

This was originally developed for use with English language at Stanford University. The tagger is based on the maximum entropy model. The improved version, which was published in 2003, adds support for other languages together with speed and usability improvements.

The latest version comes with trained models for Chinese, German and Arabic. It claims to have a 96.42% accuracy for Arabic. The tagger was trained using the training component of the Arabic Penn Treebank (ATB). It uses augmented Bies mapping of ATB tags (Toutanova et al., 2003).

2.3.5 Hidden Markov Model (HMM)

In Al Shamsi (Al Shamsi & Guessoum, 2006), the proposed Hidden Markov Model (HMM) POS tagger has been tested and achieved a performance of 97%. It used a very simple POS tag set of 55 tags. The training was performed on a special small corpus consisting of a 9.15 MB corpus of native Arabic articles. The authors used a stemmer for segmenting and separating affixes from the stem to produce prefix, stem, and suffix parts.

2.3.6 Brill (Transformation) + Morphological Analyzer

In AlGahtani (Ndie et al., 2010), transformation-based learning was used, as implemented in the Brill tagger (Brill, 1995), for POS tagging of Arabic, with segment-based tags.

They used the Buckwalter morphological analyzer (Buckwalter, 2002), and their approach was evaluated on the whole ATB as well as on ATB1. For ATB1, they achieved a POS tagging accuracy of 96.9%.

Using the whole ATB, the accuracy was 96.1%, even though large parts of the Treebank are duplicated, so that it is likely that parts of their test set were actually present in the training set (Ndie et al., 2010).

2.3.7 Rules-based and Memory-based

In Tlili-Guiassa (Tlili-Guiassa, 2006), a hybrid of rule-based and memory-based learning methods was used. Their method is based firstly on rules that are automatically learned from the training corpus (*that consider the post-position, ending of a word and patterns*) and then the anomalies were corrected by adopting a Memory-Based Learning Method (**MBL**). Secondly, by checking the exceptions to the rules, more information was made available to the learner for treating these exceptional cases. The accuracy level was 85%. The tag set was derived from that of Khoja (Khoja, 2001).

2.3.8 Statistical

In Mohamed (Mohamed & Kübler, 2010), two approaches were used. Their first approach used complex tags that described full words and did not require any word segmentation. The second approach was segmentation-based, using a segment based on machine learning. They showed that word-based POS tagging can yield better results than segment-based tagging (*93.93% vs. 93.41%*). Combining both methods resulted in a word accuracy of 94.37%. The POS tag set of the Penn Arabic Treebank was used and two sections of the ATB (**P1V3** and **P3V1**), since those two sets do not contain duplicate sentences. This data set contained approximately 500,000 words.

2.3.9 AraMorph

The Buckwalter Arabic morphological analyzer is one of the best-known Arabic morphological analysis and POS tagging systems. It was developed by LCD (*Linguistic Data Consortium*) in both Perl and Java.

The components of the Buckwalter Arabic Morphological Analyzer are the morphology analysis algorithm, and the data,

that primarily consist of three Arabic/English lexicon files: **dictPrefixes** contains 299 entries, **dictSuffixes** contains 618 entries, and **dictStems** contains 82,158 entries, representing 38,600 lemmas.

These lexicons are supplemented by three morphological compatibility tables that are used to control the **prefix-stem** combinations (*1,648 entries*), **stemsuffix** combinations (*1,285 entries*), and **prefix-suffix** combinations (*598 entries*). The algorithm of the morphology analysis and POS tagging is imbedded in the code. It uses the three lexicon files and three compatibility tables in order to perform the morphological analysis and tagging of Arabic words (Buckwalter, 2002).

3. Methodology

3.1 Introduction

Any linguistic or literary investigation of texts can benefit from computational technology. Evidently, the use of computational dictionaries, concordances and indexes, augmented by sophisticated search tools, can be extremely useful for scholars who are interested in such investigations. As a first step toward achieving this goal, we must first find a much faster and reliable system that can tag any non-vocalized Arabic text.

In this chapter, we will discuss the methodology that we applied to prove our thesis. As stated before (*in chapter one*), the proposed POS Tagger for Arabic Text is based on searching for each word in a text in our lists of Verbs and Articles. Nouns are tagged through the elimination of Verbs and Articles.

Our methodology can be described as a template-based tagging system, because every derivation of the Arabic words is produced from a template. For example, a simple sentence like (حضر زيد إلى المدرسة) can be tagged by comparing the first word (حضر) with our Articles list, which contains every Arabic article and all of their template derivations.

The word will be tagged "*Article*" if it is found. Otherwise, the same word will be compared with the Verb list, which contains every Arabic verb and all of their template derivations. The word will be tagged "*Verb*" if it is found. Otherwise, the word will be, by default, tagged as "*Noun*". These comparisons will be made for every word until they have all been tagged.

Figure 9: An example of tagging an Arabic sentence.

3.2 Verbs and Articles Lists

To apply our method, as discussed in section 1.6, we must first prepare a list of the articles and verbs of the Arabic language. The comparisons that we will make to tag a non-vocalized Arabic text must relate to whole words (includindg the *prefix, infix* and *suffix*).

Many of the prefixes and suffixes in our verbs and articles lists are generated within the derivations of the verbs and articles. However, we have added the missing prefixes through a program to each and every Verb and Article.

3.3 The Design

A methodological approach to building the model design is required in order to continue.

The methodology for this design is based on a database of Arabic verbs and articles, internal buffers, a comparison algorithm, and a raw, non-vocalized Arabic text.

First, the verbs and articles must be uploaded, and then both the Arabic verbs as well as the Arabic articles must be arranged into arrays according to their length. When the arrangement has been completed, each word of the raw, non-vocalized Arabic text is compared with both Articles and Verbs arrays to produce the resulting tagged text.

The proposed model includes the following steps:

Step-1 Uploading Articles and Verbs:

In this step, the proposed design reads the database of Articles and Verbs and stores them in the buffer. According to the length of each Article and Verb, the system will store them into arrays. For example, the verb "حضر" will be stored in the three-lettered array. (*See Figure 10*)

Less Than Two Letters	16
Two Letters	2,190
Three Letters	42,466
Four Letters	447,101
Five Letters	2,075,181
Six Letters	5,700,889
Seven Letters	12,166,765
Eight Letters	19,839,791
Nine Letters	24,240,970
Ten Letters	22,405,515
Eleven Letters	15,291,729
More Than Eleven Letters	10,360,471
Total	**112,573,084**

Figure 10: The arrays used in the buffer and the total number of elements.

Step-2 Uploading the Raw Text File:

At this point, the proposed design reads the raw, non-vocalized Arabic text file word by word and stores them in the buffer. Then, the system will identify the length of each raw word.

Step-3 Comparison Algorithm:

In this step, using a binary search algorithm, according to the length of each non-vocalized raw word, the system will compare it with the proper length array of the Articles. If a match is found, the raw, non-vocalized Arabic word will be tagged "*Article*".

Otherwise, the system will compare it with the proper length array of Verbs. If the raw word is found, it will be tagged "*Verb*".

Step-4 Tagging Nouns:

By default, if the system does not find the word from the raw Arabic text in either the Verbs and Articles lists, it will be tagged "*Noun*".

These steps are repeated as a cycle for each word in the raw Arabic text file. In Figure 11, the proposed system is shown with all of its components, followed by a complete notation of each component.

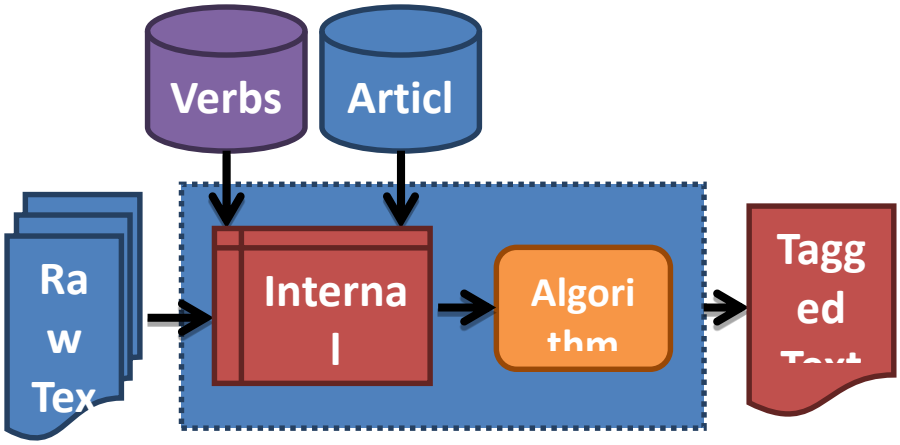

Figure 11: Our proposed approach: the Template-Based Approach to POS Tagging

Raw Text: A non-vocalized, traditional Arabic text that can be found, for example, in newspapers, books and on Arabic websites.

Verbs List: A list of all Arabic verbs and all of their derivations that we have developed by combining three-lettered verbs (*from Ba-Aziz* (Ba-Aziz, 2009)) with the four-lettered verbs that we have found.

Articles List: A list of all of the Arabic articles and all of their derivations that we have found and developed.

Internal Buffer: Different variables (*or arrays*) that are used to contain the uploaded lists of verbs and articles, which are used to compare the data according to the length of the words.

Algorithm: A binary search searches sorted lists using a divide and conquer technique. On each iteration, the search domain is halved, until the result is found. Since our Verbs and Articles lists are sorted, using a binary search algorithm is the optimal choice.

Tagged Text: The result of the proposed model. Each word of the input is tagged with one of the following tags: *Article, Verb or Noun*.

3.4 Implementation

The model will be built based on the requirements mentioned in the previous sections. That will include building a program based on the initial analysis. In the initial phases of the research, it was hoped that an existing Arabic three-lettered verbs list would be available. The verbs list was provided as a text file by the Arabic Language Template grammars' component-based technology (Ba-Aziz, 2009).

The four-lettered verbs (*about 2,000 verbs*) and the articles (*over 4,000 articles*) of the Arabic language have never been handled before. All of our data, both new and old, must be prepared for our method; we must add the proper prefixes to the whole data.

The result produced a total of over 112 million verbs and articles, which are used in our comparisons to prove our hypothesis.

3.4.1 Model Setup

The model was built using Python 3.4 language, involving the 64-bit version, on a Windows 7 64-bit platform using an Intel i7 processor with 16G bytes of memory. The system programming consisted of over 800 lines with over 10 functions.

3.4.2 Uploading the Articles and Verbs

To upload Articles and Verbs according to their length, the program first removed any duplications then sorted and stored them into arrays.

```
for item in newWordLine:
    if len(item)<2: lessArray.append(item+'\n')
    elif len(item)==2: twoArray.append(item+'\n')
    elif len(item)==3: threeArray.append(item+'\n')
    elif len(item)==4: fourArray.append(item+'\n')
    elif len(item)==5: fiveArray.append(item+'\n')
    elif len(item)==6: sixArray.append(item+'\n')
    elif len(item)==7: sevenArray.append(item+'\n')
    elif len(item)==8: eightArray.append(item+'\n')
    elif len(item)==9: nineArray.append(item+'\n')
    elif len(item)==10: tenArray.append(item+'\n')
    elif len(item)==11: elevenArray.append(item+'\n')
    elif len(item)>11: moreArray.append(item+'\n')
lessArray = del_dups(lessArray)
lessArray = sort_array(lessArray)
outfile = codecs.open("lessArray-I" + ".txt", encoding="utf-8" ,mode='w')
for word in lessArray:
    outfile.write(word)
outfile.close()
lessArray=[]
print ("Done lessArray file..")

twoArray = del_dups(twoArray)
twoArray = sort_array(twoArray)
outfile = codecs.open("twoArray-I" + ".txt", encoding="utf-8" ,mode='w')
for word in twoArray:
    outfile.write(word)
outfile.close()
twoArray=[]
print ("Done twoArray file..")
```

Figure 12: Reading the Verbs and Articles files and then removing any duplications, before sorting and storing them according to length.

3.4.3 Uploading the Raw Text File

The program now reads the raw, non-vocalized Arabic text file, word by word, and stores them in the buffer.

```
filename = "D:\Test\Quran.txt"
if filename != '':
    f1 = open(filename,'r')
    strfiletext = f1.read()
    f1.close()
```

Figure 13: Uploading the raw text file.

3.4.4 Comparison Algorithm

According to the length of each non-vocalized raw word, the program will compare it using a binary search algorithm with the proper length array of Articles and Verbs.

```
def binary_search(a, x):

    i = bisect_left(a, x)
    if i != len(a) and a[i] == x:
        return i
    #raise ValueError
    return -1
with codecs.open('Result-Report-Quran.txt', encoding='utf-8',mode='w') as outfile:
    for item in strQuran:
        if binary_search(strOrderedParticles,item)>=0:
            numArticle+=1
            outfile.write("The word ("+item+")\tFound as ARTICLE\r\n")
        elif len(item)<2 and binary_search(lessArray,item)>=0:
            numVerb+=1
            outfile.write("The word ("+item+")\tFound As VERB in lessArray \r\n")
        elif len(item)==2 and binary_search(twoArray,item)>=0:
            numVerb+=1
            outfile.write("The word ("+item+")\tFound As VERB in twoArray \r\n")
        elif len(item)==3 and binary_search(threeArray,item)>=0:
            numVerb+=1
            outfile.write("The word ("+item+")\tFound As VERB in threeArray \r\n")
        elif len(item)==4 and binary_search(fourArray,item)>=0:
            numVerb+=1
            outfile.write("The word ("+item+")\tFound As VERB in fourArray \r\n")
        elif len(item)==5 and binary_search(fiveArray,item)>=0:
```

Figure 14: Comparison Algorithm.

3.4.5 Tagging Nouns

By default, if the program did not find the word from the raw Arabic text in either the Verbs or Articles lists, it will be tagged "*Noun*".

```
elif len(item)==9 and binary_search(nineArray,item)>=0:
    numVerb+=1
    outfile.write("The word ("+item+")\tFound As VERB in nineArray \r\n")
elif len(item)==10 and binary_search(tenArray,item)>=0:
    numVerb+=1
    outfile.write("The word ("+item+")\tFound As VERB in tenArray \r\n")
elif len(item)==11 and binary_search(elevenArray,item)>=0:
    numVerb+=1
    outfile.write("The word ("+item+")\tFound As VERB in elevenArray \r\n")
elif len(item)>11 and binary_search(moreArray,item)>=0:
    numVerb+=1
    outfile.write("The word ("+item+")\tFound As VERB in moreArray \r\n")
else:
    numNoun+=1
    outfile.write("The word ("+item+")\tFound As NOUN\r\n")
```

Figure 15: Tagging nouns by default.

```
The word (بسم)      Found As VERB in threeArray
The word (الله)     Found As NOUN
The word (الرحمن)   Found As NOUN
The word (الرحيم)   Found As NOUN
The word (قل)       Found As VERB in twoArray
The word (أعوذ)     Found As VERB in fourArray
The word (برب)      Found As NOUN
The word (الناس)    Found As NOUN
The word (ملك)      Found As VERB in threeArray
The word (الناس)    Found As NOUN
The word (إله)      Found As NOUN
The word (الناس)    Found As NOUN
The word (من)       Found as ARTICLE
The word (شر)       Found As VERB in twoArray
The word (الوسواس)  Found As NOUN
The word (الخناس)   Found As NOUN
The word (الذي)     Found as ARTICLE
The word (يوسوس)    Found As NOUN
The word (في)       Found as ARTICLE
The word (صدور)     Found As NOUN
The word (الناس)    Found As NOUN
The word (من)       Found as ARTICLE
The word (الجنة)    Found As NOUN
The word (والناس)   Found As NOUN
```

Figure 16: Part of the result.

3.4.6 Python 3.4

Python is a powerful computer language that is used in a variety of applications. It emphasizes code readability, a huge standard library and, above all, it is open source. Using Python 3.4 enabled us to benefit from deploying readymade components' features.

3.5 Evaluation methods

There exist different ways of evaluating a tagger; among the most common are the accuracy or success rate.

$$Success\ rate = \frac{Correctly\ tagged\ tokens}{Total\ number\ of\ tokens} \times 100$$

The success rate is defined over all tags and evaluated by comparing the tags assigned by a tagger with each word assigned to its best tag.

4. Results and Conclusions

4.1 Introduction

In this chapter, we will discuss the results of our research, and the fulfillment of our objectives.

To evaluate our proposed method, we used pre-tagged words from "The Quranic Arabic Corpus", making a total of 78,245 words (Dukes, 2009), with our method, the Template-based tagging approach compared with (AraMorph) a rule-based tagging approach (Buckwalter, 2002) and the Stanford Log-linear Part-Of-Speech Tagger (Toutanova et al., 2003).

Human Tagged		AraMorph		Correctnes	Stanford		Correctnes	Our Method		Correctnes
الم	N	الم	V	0	الم	V	0	الم	N	1
ذلك	A	ذلك	U	U	ذلك	A	1	ذلك	N	0
الكتاب	N	الكتاب	U	U	الكتاب	N	1	الكتاب	N	1
لا	A	لا	U	U	لا	A	1	لا	A	1
ريب	N	ريب	N	1	ريب	N	1	ريب	N	1
فيه	A	فيه	A	1	فيه	N	0	فيه	A	1
هدى	N	هدى	V	0	هدى	N	1	هدى	V	0
للمتقين	N	للمتقين	A	0	للمتقين	N	1	للمتقين	N	1
الذين	A	الذين	U	U	الذين	A	1	الذين	A	1
يؤمنون	V	يؤمنون	U	U	يؤمنون	V	1	يؤمنون	V	1
بالغيب	N	بالغيب	A	0	بالغيب	N	1	بالغيب	N	1
ويقيمون	V	ويقيمون	U	U	ويقيمون	N	0	ويقيمون	V	1
الصلاة	N	الصلاة	U	U	الصلاة	N	1	الصلاة	N	1
ومما	A	ومما	U	U	ومما	N	0	ومما	N	0
رزقناهم	V	رزقناهم	V	1	رزقناهم	N	0	رزقناهم	N	0
ينفقون	V	ينفقون	U	U	ينفقون	V	1	ينفقون	V	1
والذين	A	والذين	U	U	والذين	N	0	والذين	A	1
يؤمنون	V	يؤمنون	U	U	يؤمنون	V	1	يؤمنون	V	1

Figure 17: Part of the results.

4.2 AraMorph

The Buckwalter Arabic morphological analyzer is one of the best-known Arabic morphological analysis and POS tagging systems. It was developed by LCD (*Linguistic Data Consortium*) in both Perl and Java.

The components of the Buckwalter Arabic Morphological Analyzer are the morphology analysis algorithm and the data, that primarily consist of three Arabic/English lexicon files: **dictPrefixes** contains 299 entries, **dictSuffixes** contains 618 entries, and **dictStems** contains 82,158 entries, representing 38,600 lemmas. These lexicons are supplemented by three morphological compatibility tables that are used to control **prefix-stem** combinations (1,648 entries), **stemsuffix** combinations (1,285 entries), and **prefix-suffix** combinations (598 entries). The algorithm for the morphology analysis and POS tagging is imbedded in the code. It uses the three lexicon files and the three compatibility tables in order to perform morphological analysis and the tagging of Arabic words (Buckwalter, 2002).

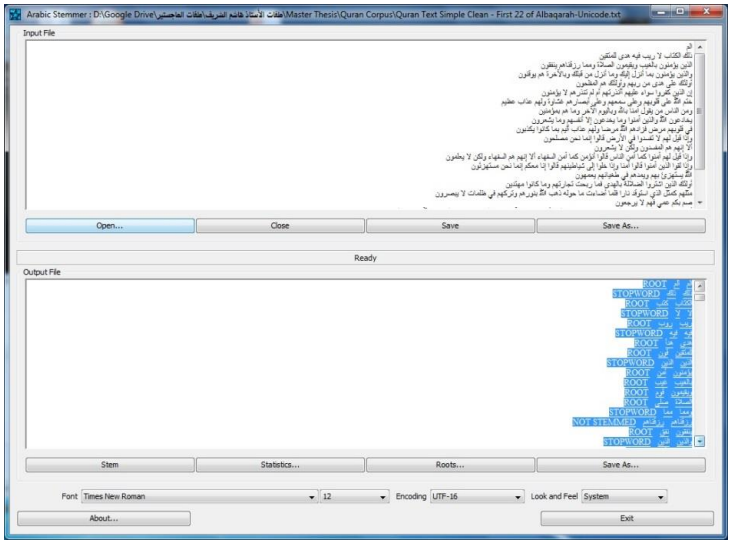

Figure 18: Screenshot of the AraMorph system

4.3 The Stanford POS Tagger

The Stanford part-of-speech tagger uses both preceding and following tag contexts via a dependency network representation, lexical features, including jointly conditioning multiple consecutive words, the effective use of priors in conditional log linear models, and the fine-grained modeling of unknown word features. By combining these ideas, the tagger gives a 97.24% accuracy level on the **Penn Treebank WSJ**, an error reduction of 4.4% on the best previous single automatically learned tagging result (Toutanova et al., 2003).

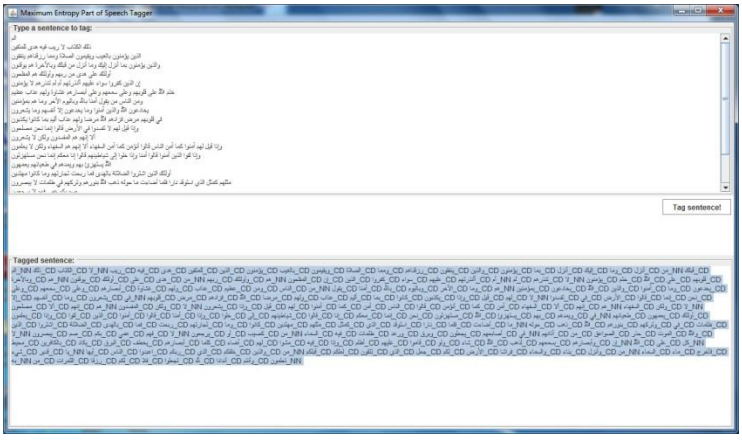

Figure 19: Screenshot of the Stanford system

4.4 The Quranic Arabic Corpus

The Quranic Arabic Corpus is an online, annotated linguistic resource with multiple layers of annotation, including morphological segmentation, part-of-speech tagging, syntactic analysis using dependency grammar and a semantic ontology (Dukes, 2009).

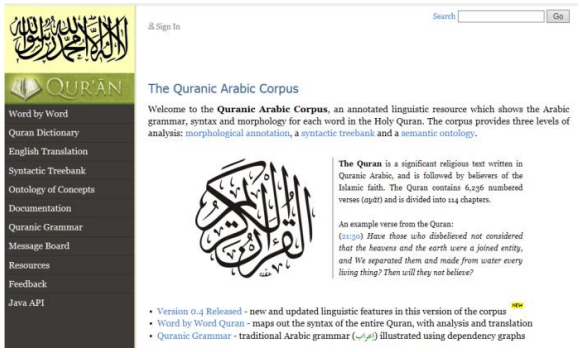

Figure 20: Screenshot of the Quranic Arabic Corpus website

4.5 Results

The AraMorph results produced 14% wrongly-tagged words and Stanford produced 32% wrongly-tagged words, while our method produced only 12% wrongly-tagged words out of all of the words in the "*The Quranic Arabic Corpus*". AraMorph tagged 46% of words as unknown words, while Stanford and our method tagged none as unknown. Finally, AraMorph produced 40% correctly-tagged words and Stanford produced 68% correctly-tagged words, while our method produced 68,501 correctly-tagged words (88%).

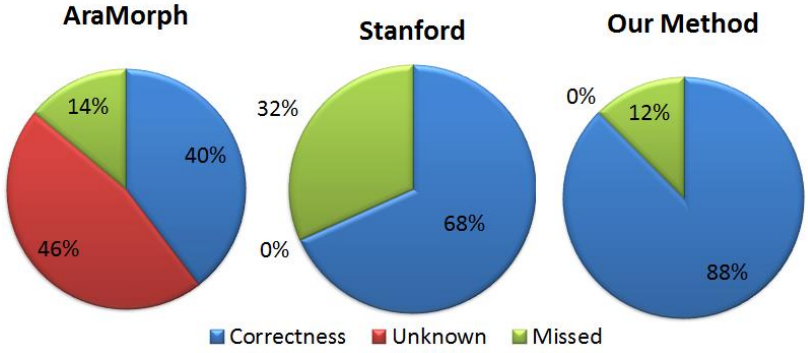

Figure 21: Proof of the validity of the methodology

	AraMorph	Stanford	Our Method
Correctness	40%	68%	88%
Unknown	46%	0%	0%
Missed	14%	32%	12%

Figure 22: The Comparison Table

4.6 Conclusions

We conclude that, because Arabic language has static verb derivations and a limited number of words, a list of all of the Verbs and Articles in Arabic language has been created. Our hypothesis states that, if the word from the text was not found in our list, it is a Noun. Therefore, Nouns are found through the elimination of Verbs and Articles.

After gathering over 112 million words (*verbs and articles*), a model was built to apply our method and test it. Our method has the highest number of correct tags compared with two other well-known Arabic POS tagging systems.

We conclude that our method, compared to AraMorph (a rule-based Arabic tagger) and the Stanford POS tagger, is far more efficient and accurate.

We have proved that the simplicity of our design, with it large-scale data comparisons, is far more effective than the complexity of the other methods.

5. References

Ndie. T.D, Tangha. C, Ekwoge. F.E. (2010). MDA (Model-Driven Architecture) as a Software Industrialization Pattern: An Approach for a Pragmatic Software Factories. *Journal of Software Engineering and Applications*, 3 (6) 561.

Chowdhury, G.G. (2003). Natural language processing, *Annual review of information science and technology*, 37, 51-89.

AlGahtani. S, Black. W, McNaught. J. (2009). Arabic part-of-speech tagging using transformation-based learning, in: Proceedings of the 2nd International *Conference on Arabic Language Resources and Tools*, Cairo, pp. 66-70.

Benson. M. (2014). Landscape: History, Present Barriers, and The Road Forward, in: *The Art of Software Thermal Management for Embedded Systems, Springer*, pp. 13-45.

Ba-Aziz. B. (2009). Arabic Language Template Grammars Component Based Technology, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia.

Jurafsky. D, Martin. J.H. (2000). Speech & Language Processing, Pearson Education India.

Zughoul. M.R, Abu-Alshaar. A.M.i. (2005). English/Arabic/English machine translation: A historical perspective, Meta: *Journal des traducteursMeta:/Translators' Journal*, 50, 1022-1041.

Wintner. S (2004). Hebrew computational linguistics: *Past and future, Artificial Intelligence Review*, 21. 113-138.

Al-Sughaiyer. I.A, Al-Kharashi. I.A. (2004). Arabic morphological analysis techniques: A comprehensive survey, *Journal of the American Society for Information Science and Technology*, 55. 189-213.

Seikaly. Z.A. (2007). The Arabic Language: The Glue That Binds the Arab World, *AMIDEAST Publications*.

Wright. W, Caspari. C.P. (2011). A grammar of the Arabic language, Cosimo, Inc.

Diab. M, Hacioglu. K, Jurafsky. D. (2004). Automatic tagging of Arabic text: From raw text to base phrase chunks, in: Proceedings of HLT-NAACL 2004: Short Papers, *Association for Computational Linguistics*, pp. 149-152.

Diab. M. (2009). Second generation AMIRA tools for Arabic processing: Fast and robust tokenization, POS tagging, and base phrase chunking, in: *2nd International Conference on Arabic Language Resources and Tools*.

Habash. N, Rambow. O. (2005). Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop, in: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, *Association for Computational Linguistics*, pp. 573-580.

Khoja. S. (2001). APT: Arabic part-of-speech tagger, in: Proceedings of the Student Workshop at NAACL, pp. 20-25.

Toutanova. K, Klein. D, Manning. C.D, Singer. Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network, in: *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*-Volume 1, Association for Computational Linguistics, pp. 173-180.

Al Shamsi. F, Guessoum. A. (2006). A hidden Markov model-based POS tagger for Arabic, *Des Journées internationales d'Analyse statistique des Données Textuelles* (JADT-8), Besançon, 31-42.

Brill. E. (1995). Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging, Computational linguistics, 21. 543-565.

Buckwalter. T. (2002). Arabic Morphological Analyzer (AraMorph), in, Version.

Tlili-Guiassa. Y. (2006). Hybrid Method for Tagging Arabic Text, *Journal of Computer science*, 2.

Mohamed. E, Kübler. S. (2010). Arabic Part of Speech Tagging, in: LREC.

Dukes. K. (2009). The Quranic Arabic Corpus.

Doi: doi.org/10.52132/Ajrsp.e.2021.32.1