A Framework to Automate the Parsing of Arabic Language Sentences

Essam Al Daoud and Abdullah Basata
Faculty of Science and Information Technology, Zarqa Private University, Jordan

Abstract: *This paper proposes a framework to automate the parsing (إعراب) of Arabic language sentences in general, although it focuses on the simple verbal sentences but it can be extended to any Arabic language sentence. The proposed system is divided into two separated phases which are lexical analysis and syntax analysis. Lexical phase analyses the words, finds its originals and roots, separates it from prefixes and suffixes, and assigns the filtered words to special tokens. Syntax analysis receives all the tokens and finds the best grammar for the given sequence of the tokens by using context free grammar. Our system assumes that the entered sentences are correct lexically and grammatically.*

Keyword: Lexical analysis, syntax analysis, Arabic language parser.

Received December 5, 2007; accepted February 20, 2007

1. Introduction

Arabic ranks fourth in the world's league table of languages, with an estimated 186 million native speakers. As the language of the Qur'an, the holy book of Islam, it is also widely used throughout the Muslim world. It belongs to the Semitic group of languages which also includes Hebrew and Amharic, the main language of Ethiopia.

Natural language analysis serves as the basic block upon which natural language applications such as machine translation, natural language interfaces, and speech processing can be built. A natural language parsing system must incorporate three components of natural language, namely, lexicon, morphology, and syntax. As Arabic is highly derivational, each component requires extensive study and exploitation of the associated linguistic characteristics. Arabic grammar is a very complex subject of study; even Arabic-speaking people nowadays are not fully familiar with the grammar of their own language. Thus, Arabic grammatical checking is a difficult task. The difficulty comes from several reasons: the first is the length of the sentence and the complex Arabic syntax, the second is the omission of diacritics (vowels) in written Arabic 'التشكيل', and the third is the free word order nature of Arabic sentence. The modern form of Arabic is called Modern Standard Arabic (MSA) [2, 5, 6]. MSA is a simplified form of classical Arabic, and follows the same grammar. The main differences between classical and MSA are that MSA has a larger (more modern) vocabulary, and does not use some of the more complicated. Arabic words are generally classified into three main categories: noun, verb and particle. While an Arabic sentence has two forms: nominal sentence and verbal sentence.

The proposed system covers the basic grammar rules for verbal sentence which can be generalized to any sentence. We will call the proposed system: A'reb (أعرب). However, A'reb has the following limitations:

- The system is assuming that sentence has been written correctly, whether morphologically or grammatically, and grammar correction is not included right now.
- As a nature of Arabic verbs, the verb could be in passive, or active voice e.g., ضرب could be read as ضُرِبَ (doreb) or ضَرَبَ (darab), the system assumes the verb as it is in the active voice.
- The A'reb does not prevent errors that are related to incorrect use of semantic meaning, means that the semantic analysis is not verified.

The goals of the project which we like to achieve in our A'reb system are:

- To serve the Arabic in the automation field, especially in noteworthy subject like E'rab.
- To build kernel functions, which can be used to Arabic sentence correction, translation, natural language interfaces, and speech processing.
- To design a system that applies the major of lexical services , like getting the root, the various form of the word,
- To design a comprehensive system that covers the most verbal sentence cases, including repetition case.
- To design an easy to use and intuitive system with short learning curve.

- To design a general system to be applicable for different persons such as student or teachers.
- To provide an e-learning notion in the simplest way.

2. The Architecture of A'reb

The system is based on syntactic analysis and relies on a feature relaxation approach for detection of ill-formed Arabic sentences. A'reb helps the user to write a sentence by analyzing each word and then only accepting the sentence if it is grammatically correct. The main features of our A'reb system are: give some lexical feature of Arabic words and parse the simple verbal Arabic language sentences, but it can be extended easily to any Arabic language sentence. The design of the whole system is shown in Figure 1. The A'reb is basically composed of two parts: An Arabic lexical analyzer, and a syntax analyzer.

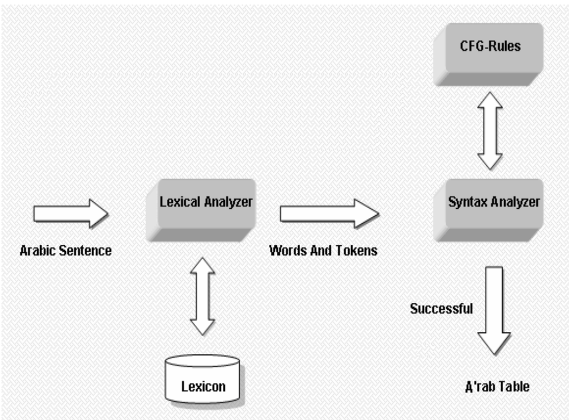

Figure 1. The Architecture of the A'reb system.

With quick looking to the system main functions, it is evident that the system needs only two stakeholders: user and administrator. The administrator tasks are updating the data, and adding more services.

3. Lexical Analysis

The main function of a lexical analyzer is to break down the input stream into lexical items or morphemes. If the morpheme can function alone, such as the word مهندس (engineer), it is called a free morpheme. Other morphemes cannot be used by themselves, such as the general plural ending ون and the letters ين in the word مهندسون (engineers). Such morphemes are called ‘bound’. Bound morphemes, in Arabic, serve as additions at the beginning or ending of a stem. Using the definitions of free and bound morphemes, a word can be defined as a single free morpheme, and an inflected word can be defined as a complex form which is a single free morpheme combined with one or more bound morphemes [3, 7].

A lexical (morphological) analysis must tokenize and categorize the Arabic words (past, present, future, intransitive, transitive…) and separate them from prefixes and suffixes. In previous works, many methodologies have suggested to drive all the Arabic words from the roots. However, the best algorithm suggested has an accuracy of less than 90% which is not accepted in Arabic sentence parsing [1, 2, 8]. Thus in A’reb system we must store all the Arabic words in a database (lexicon) excluding the prefixes and the suffixes (which is around 2 millions words), and by using tree indexing we can find the required word very fast O(s), where s is the length of the required word, and since the maximum length of the Arabic word can not be more than 10, thus the complexity is constant.

In our database or lexicon we have used five main tables namely root, present, order, noun and particle table. All the tables' entries are free morpheme (without prefixes and suffixes).

Two main tasks must be achieved in the A’reb lexical analysis: the first is to separate the input words from the prefixes and suffixes and the second is to assign a suitable symbol to each lexeme. To separate the Arabic word from prefixes and suffixes we suggest a multi-level comparing as follows:

- First Level: the input words without prefixes and suffixes, which means comparing the input word with the word stored in the database directly, and then tokenizing it. If the word is not in the database then we go to the second level.
- Second Level: the input word without prefixes, which means that we have to isolate all the possible suffixes and then go to the first level. If the word is not in the database then we go to the third level.
- Third Level: the input words without suffixes, which means that we have to isolate all the possible prefixes and then we go to the first level. If the word is not in the database then we go to the fourth level.
- Fourth Level: the input words with prefixes and suffixes, which means that we have to isolate all the possible prefixes and suffixes and then we go to the first level. if the word is not in the database then we consider it a noun or we ask the user.

Table 1. Examples explaining the separation of the input words.

<table>
<tr><th colspan="4">After Separation</th><th>Input Word</th></tr>
<tr><td colspan="2">يسبق</td><td colspan="2">سـ</td><td>سيسبق</td></tr>
<tr><td>هم</td><td>و</td><td>سبق</td><td>فـ</td><td>فسبقوهم</td></tr>
<tr><td>كم</td><td>يكفي</td><td>س</td><td>ف</td><td>فسيكفيكم</td></tr>
<tr><td>ان</td><td colspan="2">طفل</td><td>الـ</td><td>الطفلان</td></tr>
</table>

The second lexical analysis task is to assign a suitable symbol to each lexeme. To achieve this task we first have to suggest a symbol (token) to each group of the lexemes, where each group has a common parsing behavior, Table 2 contains sample of the suggested symbols (tokens). Table 4 explains the output stream after the lexical analysis achieved on a stream of Arabic sentences.

Table 2. Sample of the suggested symbols (tokens).

Lexemes/Words	**Symbol Token Terminal**	**Notes**
A word in the Noun Table	N	الاسم
A word in the Root Table with transitive attribute	ls	الفعل الماضي المتعدي
A word in the Root Table with intransitive attribute	lm	الفعل الماضي اللازم
A word in the Order Table with transitive attribute	om	الفعل الأمر المتعدي
A word in the Order Table with intransitive attribute	ol	الفعل الأمر اللازم
A word in the Present Table with transitive attribute	prM	الفعل المضارع المتعدي
A word in the Present Table with intransitive attribute	prL	الفعل المضارع اللازم
كهن,كهم,هن,هم,هما,ها,ه	Swh	هاء الغائب، الغائبة ...
تموا,تن,تم,تما,ت	St	تاء المخاطب، المخاطبة...
نا	Na	نون المتكلمين
ن	Snn	نون النسوة
ا	Sa	ألف الاثنين
و	Sw	واو الجماعة
وا	Swa	واو الجماعة
(ي) تأتي مع المؤنث سواء كانت مخاطبة أو غائبة	Sy	مع الأفعال الخمسة
الذي,التي,اللائي,اللاتي	Pcm	الأسماء الموصولة المبنية
اللذان, اللتان	Pcba	الأسماء الموصولة المعربة
الذين,اللتين	Pcby	الأسماء الموصولة المعربة2
هذا,هذه,تلك,ذلك,هؤلاء,أولئك,أولاء	Pim	أسماء الإشارة المبنية
هذان,هاتان,ذانك,تانك	Pira	أسماء الإشارة المعربة
هذين,هاتين	Piry	أسماء الإشارة المعربة2
أنت,هو,هي,هما,هن,أنتن	Pff	الضمائر المنفصلة
نحن	Pfd	الضمائر المنفصلة2
أنا,أنتما,أنتم,هما,هم	Pfs	الضمائرالمنفصلة3
ف,و, ثم....	PreAtf	أحرف العطف
ين	Sfvy	مع الأفعال الخمسة
ان	Sfva	مع الأفعال الخمسة
ون	Sfvw	مع الأفعال الخمسة
و	PreAtf	حرف عطف
ف	PreAtf	حرف عطف
ال	Al	الألف واللام
أين,من,أنى ,كيف,كم, هل	PreI	أحرف الاستفهام
إذا,إذ,لولا,لو,إذما,حيثما,أينما,مهما	PreSH	أحرف الشرط
قد,إن	PreK	أحرف التأكيد
نعم,بلى	PreJ	أحرف الجواب
لا,ما	PreN	أحرف النفي
إذن,كي,لن,أن	PreNsb	أحرف النصب
لا,لما, لم	PreJzm	أحرف الجزم
لولا,هلا,لوما, ألا	PreD	أحرف التحضيض

Ambiguity is another problem that must be solved during the lexical analysis. The ambiguity in the Arabic words occurs if we do not use the diacritics التشكيل as the following examples:

فهمت الدرس

سبقنا الحصان

The first example has three meanings without diacritics:

فهمْتَ الدرس، فهمْتِ الدرس، فهمَتُ الدرس

The second example has two meanings without diacritics:

سبقْنا الحصانَ، سبقَنا الحصانُ

The ambiguity problem can be solved by two ways; the first is asking the user each time the ambiguity occur and the second is accepting, parsing and displaying all the possibilities.

4. Syntax Analysis

Parsing (more formally syntactical analysis) is the process of analyzing a sequence of tokens to determine its grammatical structure with respect to a given formal grammar, the parsing transforms input text into a data structure, usually a tree, which is suitable for later processing and which captures the implied hierarchy of the input [4].

There are two tasks in the syntax analysis phase that must be accomplished, the first is determining all the Arabic language rules and then write the equivalent Context Free Grammar (CFG). The second is choosing and building the parser, in the proposed system we have selected the recursive parser.

There are two possible output of the syntax analyzer: first; the analysis is successful and no syntactic inconsistencies are found, in this case the sentence will be able to parse and the result (E'rab) will printed. Second; the analysis fails, and the results contain at least one syntactic inconsistency. In this case an error message is displayed and the system will ask the user to correct the errors. Moreover, the system can advise the user about the nearest correct sentence.

5. Arabic Language Context Free Grammar

A grammar is a formal system which specifies which sequences of words are well-formed in the language, and which provides one or more phrase structures for well-formed sequences.

Table 3. CFG non-terminals.

Non-Terminal	**Meaning**
AT	Arabic Text
NS	Nominal Sentence
VS	Verbal Sentence
SUB	Subject
O	Object
V	Verb
PRE	Prefixes

The CFG consists of four components: set of terminals, set of non- terminals, a start symbol and set of productions. The terminals in the proposed system are the set of all tokens received from the lexical analyzer and explained in Table 2, while the non-terminals are the set in Table 3.

Table 4. Output lexemes and tokens of input sentences.

After Lexical Analysis							Input Sentence			
Tokens		Lexemes								
N	Al	Swa	prM	PreJzm	طعام	ال	وا	يأكل	لم	لم يأكلوا الطعام
Tr	N	Al	lm	ان	مهندس		ال	جاء		جاء المهندسان

The start production is AT → VS AT|NS AT|ε and the suggested productions of past verb intransitive are:

VS → lm SUB | PRE lm SUB

PRE → preAtf | preK |preSH | preI | preN | preJ | preD

SUB → es SUB2 | em SUB2 | er SUB2 | ef SUB2 | sa SUB2 | sw SUB2 | snn SUB2 | st SUB2 | N SUB2 | pim SUB2 | pira SUB2 | pcm SUB2 | pcba SUB2 | pcby SUB2 | pf SUB2 | dm SUB2 | piry SUB2 | pff SUB2 | pfd SUB2 | pfs SUB2

SUB2 → preAtf es SUB2 | preAtf em SUB2 | preAtf er SUB2 | preAtf ef SUB2 | preAtf N SUB2 | preAtf pim SUB2 | preAtf pira SUB2 | preAtf pcm SUB2 | preAtf pcba SUB2 | preAtf pcby SUB2 | preAtf pf SUB2 | preAtf dm SUB2 | preAtf piry SUB2 | preAtf pff SUB2 | preAtf pfd SUB2 | preAtf pfs SUB2 | ε

the suggested productions of present verb transitive are:

VS → prM SUB O | PRE1 prM SUB O | prM SUB O | PRE2 prM SUB O | prM O SUB | PRE1 prM O SUB | O prM SUB | PRE2 O prM SUB| PRE3 prM SUB O| PRE3 O prM SUB ...

PRE1→ preK
PRE2→ preNsb
PRE3 → preJzm

--
SUB → es SUB2 | em SUB2 | er SUB2 | ef SUB2 | sa SUB2 | sw SUB2 | snn SUB2 | sy SUB2 | N SUB2 | pim SUB2 | pira SUB2 | pcm SUB2 | pcba SUB2 | pcby SUB2 | pf SUB2 | dm SUB2 | piry SUB2 | pff SUB2 | pfd SUB2 | pfs SUB2 | sfva SUB2 | sfvw SUB2 | sfvy SUB2

SUB2 → preAtf es SUB2 | preAtf em SUB2 | preAtf er SUB2 | preAtf ef SUB2 | preAtf N SUB2 | preAtf pim SUB2 | preAtf pira SUB2 | preAtf pcm SUB2 | preAtf pcba SUB2 | preAtf pcby SUB2 | preAtf pf SUB2 | preAtf dm SUB2 | preAtf piry SUB2 | preAtf pff SUB2 | preAtf pfd SUB2 | preAtf pfs SUB2 | ε

O → N O2 | es O2 | em O2 | er O2 | ef O2 | swh O2
O2 → N | es | em |ef | er | ε

And so on, we have to produce productions corresponding to all Arabic rules. Note that, we can reduce the above productions, but we have included the redundancy in the above CFG to explain our idea.

6. The Recursive Parser

A recursive parser is a top-down parser built from a set of mutually-recursive procedures where each such procedure usually implements one of the production rules of the grammar. Thus the structure of the resulting program closely mirrors that of the grammar it recognizes. The following is a part of a recursive parser algorithm which we have used:

```
Procedure AT( )
  Begin
   If  look_Ahead ={ preAtf | preK |preSH | preI | preN
   | preJ | preD | preNsb | preJzm| lm| prM }
        Call VS()
        Call AT()
  Else if  look_Ahead={ N|Pcm |Pcba |Pcby |Pim| pira.....}
        Call NS()
        Call AT()
  End Procedure

Procedure VS()
  Begin
     If  look_Ahead =lm
           Match (lm);Print "فعل ماضي مبني على الفتح "
           Call SUB()
    Else if look_Ahead =preAtf
         Match(preAtf);  Print "حرف عطف"
           Match(lm); Print "فعل ماضي مبني على الفتح"
           Call SUB()
   Else if look_Ahead = preK
           Match(preK);  Print " حرف تأكيد"
           Match(lm); Print "فعل ماضي مبني على الفتح "
           Call SUB()
  Else if look_Ahead = preSH
          Match(preK);  Print "حرف شرط"
           Match(lm); Print " فعل ماضي مبني على الفتح "
           Call SUB()
  Else
        Error
End Procedure
```

The complexity of the syntax analyzer (the recursive parser) is O(l) where l is the syntax length. Thus, the total complexity of the suggested system is O(s) + O(l) which can be performed in milliseconds.

Figure 2. Sample input/output of our system.

7. Conclusion

An Arabic parsing program is a complex program that needs extensive research and linguistic resources. In the proposed system we tried to highlight the most attractive property in Arabic language, which is Al-E'rab. However, the proposed system still needs a lot of work such as the rest of verbal sentences, nominal sentences and semantic analysis. The semantic analysis can be used to solve some type of ambiguity automatically. Once the Arabic parser is completed many problems can be solved such as automatic diacritics, Arabic sentences correction and accurate translation.

References

[1] Ahmed A. and Khaled F., "Lexical Analysis of Inflected Arabic Words Using Exhaustive Search of an Augmented Transition Network Software," *Computer Journal Practice and Experience*, vol. 23, no. 6, pp. 567-588, 1993.

[2] Black W. and El-Kateb S., "A Prototype English-Arabic Dictionary Based on Word Net," *in Proceedings of the Second International WordNet Conference GWC 2004, Czech Republic,* UK, pp. 169-174, 2004.

[3] Eman O., Khaled F., and Ahmed R., "A Chart Parser for Analyzing Modern Standard Arabic Sentence," *MT Summit'IX Workshop: Machine Translation for Semitic Languages*, USA, 2003.

[4] George L., *Artificial Intelligent, Structure,s and Strategies for Complex Problem Solving*, Benjamin-Cummings Publishing, UK, 1993.

[5] Khaled F., "Arabic GramCheck: A Grammar Checker for Arabic, Softw," *Computer Journal Practice and Experience,* vol. 35, no. 7, pp. 643-665, 2005.

[6] Khaled F., Ahmed A., Azza A., and Hoda B., "Machine Translation of English Noun Phrases into Arabic," *International Journal Computer Processing Oriental Language*, vol. 17, no. 2, pp. 121-134, 2004.

[7] Plant R. and Murrell S., "A Natural Language Help System Shell Through Functional Programming," *Computer Journal Knowledge-Based Systems,* vol. 18, no. 1, pp. 3-19, 2005.

[8] Tahir Y. and Chenfour N., "Realization of a Morphological Analyzer for Arabic Language Text," *in Proceedings of the Workshop on Information Technology, Rabat,* Morocco, pp. 348-350, 2003.

Essam Al Daoud received his BSc from Mu'tah University, MSc from Al Al-Bayt University, and his PhD in computer science from University Putra Malaysia in 2002. Currently, he is an assistant professor in the Computer Science Department at Zarqa Private University, Jordan. His research interests include quantum computation, quantum information, cryptography, natural language processing, and nanotechnology.

Abdullah Basata received his BSc from Zarqa Private University in 2007. His research interests include natural language processing, Arabic language processing, and artificial intelligent.