Towards An Arabic Parser for Modern Scientific Text

Khaled Shaalan*, Ahmed Farouk**, Ahmed Rafea*

*Computer Science Dept., Faculty of Computers and Information, Cairo Univ.
**Institute of Statistical Studies and Research (ISSR), Cairo Univ.
5 Tharwat St., Orman, Giza, Egypt.

E-mail: shaalan@esis.claes.sci.eg, a_fsadek@hotmail.com, rafea@esis.claes.sci.eg

Abstract— The present work reports our attempt in developing an Arabic Parser for modern scientific text. The parser is written in Definite Clause Grammar (DCG) and is targeted to be part of a machine translation system. The developing of the parser was a two-step process. In the first step, we acquired the rules that constitute a grammar for Arabic that gives a precise account of what it is for a sentence to be grammatical. The grammar covers a text from the domain of the agricultural extension documents. The second step was to implement the parser that assigns grammatical structure onto input sentence. Experiment on real extension document was performed. The paper will also describe our experience with the developed parser and results of its application on a real agricultural extension document.

Keywords—Parsing, Definite Clause Grammar, natural language processing

1. Introduction

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 (Khayat, 1996).

Recently, research work and development in natural language processing (NLP) systems for Indo-European languages has explored specific approaches to parsing in some depth, has consolidated practical experience, and has emphasized some trends, for example towards, lexical function grammars and deterministic parsing, and towards closer integration of syntax and semantics. This has led automatic NLP to gain in strength and self-confidence, as it has been clearly shown that working systems can be built, even if they are only very modest ones. On the contrary, little work has been done in developing parsers involving Arabic (Feddag, 1992).

An important consideration, in the development of an Arabic parsing system, is the matter of Arabic morphology. Most of the researches in Arabic NLP systems mainly concentrated on the fields of morphological analysis. In a previous work (Rafea and Shaalan, 1993), we analyzed and discussed the problem of implementing a morphological analyzer for inflected Arabic words. With respect to the implementation of the Arabic parser, we took the advantage of the already developed morphological analyzer by integrating it with the Arabic parser.
Parsing Arabic sentences is a difficult task. The difficulty comes from several sources. One is that sentences are long and complex. The average length of a sentence is 20 to 30 words, and it often exceeds 100 words. Another difficulty comes from the sentence structure. The Arabic sentence is complex and syntactically ambiguous due to the frequent usage of grammatical relations, order of words and phrases, conjunctions, and other constructions. The main goal was to implement a computer system to parse Arabic sentences. The architecture of the system is given in Figure 1. In this overview of the architecture, the arrows indicate the flow of information. Boxes are modules of the system.

This article describes our attempt in developing an Arabic Parser for modern scientific text. The next section briefly presents the acquisition of the rules that constitute a grammar for Arabic that gives a precise account of what it is for a sentence to be grammatical. The grammar covers a text from the domain of the agricultural extension documents. Then the focus turns to a description of implementation of the Arabic parser that assigns grammatical structure onto input sentence, which is the concern of this study. Next, we discuss the outcome of applying the experiment on real extension document and present its results. In a concluding section, we present some final remarks.

2. The Grammar
The grammar for Arabic contains the grammar knowledge required to analyze a modern scientific sentence. Ideally, in order to construct such a grammar one would take a general-purpose computational grammar of Arabic, and adapt it to the current domain and application. Unfortunately, however, we do not know of any computational grammar for Arabic that could easily be adapted to the present task.
The grammar is being developed especially for the purposes of understanding scientific Arabic text. On the one hand this has the advantage that the grammar can be tailored to the specific requirements of the scientific domain. On the other hand, we want to adopt general solutions as much as possible, as this increases the chances that the grammar can be used in other domains as well. Thus, in designing the grammar we seek a balance between short-term goals (a grammar which covers sentences typical for the scientific domain and is reasonably robust and efficient) and long-term goals (a grammar which covers the major constructions of Arabic in a general way).

The grammar currently covers sentences that fall into one of two categories: simple sentence and compound sentence. A simple sentence is a sentence, which is not connected by any means with another sentence. However, it may embed another sentence. A compound sentence is more than a simple sentence connected with a conjunction article (تَهْمَدِيَال/تَهْمِيَال/أَيْنِي). Simple sentences are classified into three classes: nominal sentence, verbal sentence, and special sentences. Special sentences are either the special verbs (كان و أخوانا/و أخوانا) or special particles ('يَنَأ and its sisters وَانَأ و أُخْوَانَا).

2.1 *The grammar formalism*
From a linguistic perspective, the current grammar can be characterized as a constraint based grammar, which makes heavy use of lexical information. The design of the grammar was inspired to a certain extent by Head-Driven Phrase Structure Grammar (HPSG) (Pollard and Sag, 1994).

The grammar formalism is essentially equivalent to Definite Clause Grammar (DCG) (Pereira, Shieber, and David, 1986). The choice for DCG is motivated by the fact that this formalism provides a balance between computational efficiency and linguistic expressiveness, and the fact that it is closely related to constraint-based grammar formalisms (Bouma et al., 1996), such as HPSG, Unification Grammar (Shieber, 1986), and Lexical Function Grammar (Neidle, 1994 and Sadler, 1996). Another important reason to choose DCG instead of a more restricted formalism such as context-free grammar, is the fact that DCG allows the kind of integration of syntax and semantics that is standard in constraint-based formalisms such as HPSG.

2.2 *Feature-Structures*
Grammar rules consist of a context-free skeleton to which feature-constraint are added. The context-free skeleton is important, as it ensures a reasonable level processing efficiency and facilitates experimentation with different parsing techniques (Bouma et al., 1996). The central formal operation in constraint-based grammar formalisms is unification of feature-structures. During the construction of the Arabic parser, feature-structures are translated into Prolog terms. Because of this translation step parsing can make use of Prolog’s built-in term-unification, instead of the more expensive feature-unification.

The following table describes the features used in the current grammar along with their possible values:
### Feature Possible Values

<table>
<thead>
<tr>
<th>Feature</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gender</td>
<td>male, female</td>
</tr>
<tr>
<td>Number</td>
<td>singular, dual, plural</td>
</tr>
<tr>
<td>Definiteness</td>
<td>definite, indefinite</td>
</tr>
<tr>
<td>Special Noun</td>
<td>type of special nouns</td>
</tr>
<tr>
<td>Pattern</td>
<td>form of morphological pattern such as</td>
</tr>
<tr>
<td>End Case</td>
<td>accusative, nominative, genitive</td>
</tr>
<tr>
<td>Verb</td>
<td>transitive, intransitive</td>
</tr>
<tr>
<td>Special verb</td>
<td>type of special verbs</td>
</tr>
<tr>
<td>Verb affix</td>
<td>affixes of an inflected word</td>
</tr>
<tr>
<td>Last Category</td>
<td>category of current grammatical symbol</td>
</tr>
<tr>
<td>First Category</td>
<td>category of next grammatical symbol</td>
</tr>
<tr>
<td>Noun as adjective</td>
<td>yes, no</td>
</tr>
<tr>
<td>Noun as annexation</td>
<td>yes, no</td>
</tr>
</tbody>
</table>

#### 3. Parsing

Logic programming plays an essential role in natural language analysis process because it attempts to use logic to express grammar rules and to formalize the process of parsing (Gazdar and Mellish, 1990). A grammar specified this way is known as logic grammar since it represents rules as Horn clauses (Dougherty, 1994). Logic grammars can be conveniently implemented in Prolog. Prolog-based grammars can be quite efficient in practice (Allen, 1995). Prolog interpretation algorithm uses exactly the same search strategy as the depth-first top-down parsing algorithm, so all that is needed is a way to reformulate grammar rules as clauses in Prolog. For more than a decade, definite clause grammars (DCGs) notation was developed as a result of research in natural language parsing and understanding (Pereira, Shieber, and David, 1986). DCGs allow one to write grammar rules directly in Prolog, producing a simple recursive decent parser. Prologs that conform to Edinburgh standard have DCGs as a part of their implementations. In the current system, grammar rules of Arabic are written in DCG formalism, which are translated into an executable code in Prolog. In the following subsections, we discuss the problems of specifying context-dependency and resolving ambiguities.

#### 3.1 Handling Context-Dependency

Our Arabic parser provides the means to include context-dependency in a grammar. We used this phenomenon in handling the agreement of number, gender, and definiteness, among others, between parts of a sentence. For example, the following is a grammar rule of a *verbal sentence*:

\[
\text{simple\_verbal\_sentence} \rightarrow \text{verb+agent+complementary+direct-object+complementary}
\]

This syntactic form specifies a sequence of the grammatical symbols, either terminals or non-terminals, which must occur in order to parse the input sentence. This form requires that there is an agreement between the gender of the agent and the verb of the sentence. This is handled in our parser by the unification of feature-structure, as follows.

\[
\text{verbal\_sentence}(\ldots) \rightarrow \\
\text{verb}(\ldots, \text{GENDER}, \ldots, \ldots, \ldots), \\
\text{subject}(\ldots, \text{GENDER}, \ldots, \ldots, \ldots), \\
\text{complementary}(\ldots), \\
\text{direct\_object}(\ldots), \\
\text{complementary}(\ldots)
\]


When parsing the following sentences, the first sentence is accepted because the gender feature agrees (unifies) with the agent (الفاعل) and the verb (فعل). Whereas, they do not agree in the second sentence.

"فأثرت الفلاحون في الغرب الفصح بمفردات كبيرة"

"فأثرت الفلاحون في الغرب الفصح بمفردات كبيرة"

Another type of context-dependency is when we would request from the parser to recognize a two contiguous indeterminate (نكرة) as distinguish constituent (مكتبة) such that the first indeterminate should has its morphological pattern (الاسم) as the pattern (الاسم). This is handled in our parser by the use of the unification of feature-structure, as follows.

```
distinguish_constituent (.........)-->
    indeterminate(..., ..., ..., MorphPattern1, ..., ...),
    indeterminate(..., ..., ..., MorphPattern2, ..., ...),
    {
        MorphPattern1 = "المثل"
    }.
```

So, the following fragments of a sentence are recognized as distinguish constituents.

"أفضل طريقة" "أشهر خنجر" 

3.2 Disambiguation
Our Arabic parser allows disambiguating its output in certain situations. This is discussed in the following subsections.

3.2.1 Connection Degree
Sometimes we need to enable or disable some features in the grammar in order to disambiguate sentences during parsing as early as possible. For example, if we try to analyze the sentence

"انظر إلى أوراق النمرة عند سافيا : جندها"

With the following grammar

```
verbal_sentence(...) -->
    verb(...),
    subject(...),
    complementary(...).

complementary(...) -->
    annexation_constituent(...),
    connector(...),
    annexation_constituent(...).
```

```
annexation_constituent(...) --> indeterminate(...), defined(...).
annexation_constituent(...) --> indeterminate(...), connected_pronoun (...).
annexation_constituent(...) --> accusative_time_or_place(...),
annexation_constituent(...).
annexation_constituent(...) -->
annexation_constituent(...),
connector(...),
annexation_constituent(...).

We obtain the following parse trees for the complementary of the sentence:
The disambiguation arises because the second rule of the annexation_constituent does a redundant work in recognizing the connection cases when it is called from a complementary rule. To disambiguate these trees, we have to eliminate the other two trails. This can be done by setting an explicit feature—acts as a constraint—that disambiguates the connection case of the annexation rules. Consequently, the complementary rule is defined as

```
complementary(...) -->
    annexation_constituent(…, no_connection),
    connector(…),
    annexation_constituent(…, no_connection).
```

```
complementary(...) -->
    annexation_constituent(…, no_connection),
    connector(…),
    complementary(...).
```

The result will be only one tree, the first one.

### 3.2.2 Decomposition Rejection

Sometimes we obtain undesired parse trees, because the parser decomposes a single grammatical constituent into some other separated ungrammatical constituents. For example, consider the following sentence, and its corresponding grammar:

```
"تُعَدُّ مِنِ الْمَصَّوَاتِ الْمُهْمَّةِ"

verbal_sentence(...) -->
    verb(...),
    complementary(...),
    agent(...).
```

```
verbal_sentence(...) -->
    verb(...),
    complementary(...).
```

```
complementary(...) -->
    quasi-proposition(…).
```

```
quasi-proposition(...) -->
    preposition(...),
    defined(...).
```

```
quasi-proposition(...) -->
    preposition(...),
    adjective_constituent(...).
```

```
adjective_constituent(...) -->
    defined(...),
    defined(...).
```

In analyzing this sentence, we obtain the following two parse trees:
This ambiguation problem comes from the recognition of the defined noun (ال مهمة) as set apart from the defined noun (الموضعات), where they should altogether be drawn directly from the same grammatical entity. This is resolved during parsing by adding a disambiguation rule that handles this undesirable situation. The rule should prevent the generation of the parse tree when it is possible to draw an incorrect parse fragment. Applying this solution to the verb_sentence rule given above, we rewrite it as follows:

```
verbal_sentence(...) -->
    verb(...),
    complementary(..., LastUnitInfo1, FirstUnitInfo1),
    agent(..., LastUnitInfo2, FirstUnitInfo2),
    {
        decompose_validate(LastUnitInfo1, FirstUnitInfo2).
    }.
```

All we need is to implement the prolog predicate that validates the decomposition rejection, this predicate matches all the cases of two contiguous parse fragments that must be rejected. The following disambiguation rule handles the case of (معرّفة + معرّفة) of decomposition rejection.

```
decompose_validate(LastUnitInfo, FirstUnitInfo) :-
    {
        (is_unit_type(LastUnitInfo, 'معرّفة'),
         unite_adverbed(FirstUnitInfo)) ->
        reject_type(FirstUnitInfo, 'معرّفة')
    }.
```
4. Experimental results

The purpose of our experiment was to investigate whether the parser is sufficiently robust for application to real-world Arabic text. We selected an unrestricted Arabic text, which is an agricultural extension document. The document is entitled “نشرة رقم 420 - نشرة زراعية من مركز البحوث الزراعية”, which carries many information and instructions to the Egyptian farmers. The text includes 10 pages, each of which contains about 20 sentences in average.

The implemented parser is a recursive descent parser that implicitly uses the call-stack of the language. This means that the backtracking time differs significantly with different architectures.

We have implemented the parser on a Pentium 133 processor, with 69 MByte RAM, and 2.5 GB Hard Disk. The software used was SICStus Prolog ver. 3.7.

In our discussion of the testing results we first look at whether the input sentence is parsable and then, if so, whether the complete parse is ambiguous.

The first question to ask is how good the output of the parser is. Table 1 shows the results of the parser across the document. The results falls into two categories: the parsable sentence and the unparsable sentence.

The parsable sentence category refers to the case when the parser can parse input sentences leading to one or more parse trees. Sometimes the parser only assigns incorrect grammatical structure to the input sentence. For example, the input sentence

```
/wawfinal/meemmedial/noonmedial/laminitial/alefisolated /tehmarbutafinal/fehmedial/yehmedial/ainmedial/dadinitial ... /lamwithyehisolated/alefwithhamzabelowisolated /lamfinal/yehmedial/tehwithmeeminitial ...
```

is assigned the incorrect parse tree

```
/tehmarbutafinal/tatweel /tatweel/yehmedial/lammedial/ainmedial/fehinitial /tehmarbutafinal/tatweel ... /tatweel/aininitial/aleffinal/fehinitial ) /behfinal/tatweel /tatweel/yehmedial/kafinitial/rehfinal/tehinitial /fehwithyehisolated/aleffinal/dadinitial/alefwithhamzabelowisolated )/tehmarbutaisolated/rehfinal/kafmedial/nooninitial ... )/wawfinal/meemmedial/noonmedial/laminitial/alefisolated /rehfinal/kafinitial/thalfinal/meeminitial (((((
```

Linguists analyze this sentence as if it were written as

```
```

We get this incorrect parsing because the parser fails to recognize the absence of the agent of the verbal sentence. In this case it only recognizes the annexation constituent — as the agent of the verbal sentence.

For this reason, we classified the parsable sentence category into two subcategories:

1. **Grammatical Successful.** Which has led to a complete successful parse of the input sentence.
2. **Grammatical Failure.** Which has led to a grammatically incorrect structure of the input sentence.

The unparsable sentence category refers to the case when the parser fails to parse the input sentences. Sometimes failures are due to ill-formedness of the input sentence which, is also not recognizable by linguists according to Arabic grammar rules. This unexpected syntactic phenomenon in real-world texts is beyond the parser coverage.
In other words, if the linguistic parsing fails, our parser fails. For example, the input sentence

\[
/\text{seenisolated}/wawfinal/behinitial/dalfinal/laminitial/alefisolated /seenisolated/alefwithhamzaaboveisolated/rehisolated ... /tahfinal/qafmedial/nooninitial /aleffinal/behwithhehinitial /rehfinal/hehmedial/zahmedial/yehinitial .
\]

is not parsed by our parser because the agent of the sentence—(\text{hamzaisolated/alefisolated/dalisolated/wawfinal/tatweel/seeninitial /tahfinal/tatweel/qafmedial/nooninitial})—takes the feature gender as female, but the prefix (\text{yehisolated}) of the verb (\text{rehfinal/tatweel/hehmedial/zahmedial/yehinitial}) of the sentence indicates that this feature value is expected to be male. The grammatically correct sentence would be as follows

\[
/\text{seenisolated}/wawfinal/behinitial/dalfinal/laminitial/alefisolated /seenisolated/alefwithhamzaaboveisolated/rehisolated ... /tahfinal/qafmedial/nooninitial /aleffinal/behwithhehinitial /rehfinal/hehmedial/zahmedial/tehinitial .
\]

For this reason, we classified the unparsable sentence category into two subcategories:

1. \textit{Ill-formedness}. Which has failed to parse due to a grammatically incorrect input sentence.
2. \textit{Failure}. Which has failed to assign a parse tree to a grammatically correct sentence.

\begin{table}[h]
\begin{center}
\begin{tabular}{|c|c|c|}
\hline
\textbf{Parsable} & \textbf{Grammatical Success} & 146 \ 68.8 \ \\
& \textbf{Grammatical Failure} & 44 \ 20.7 \ \\
\hline
\textbf{Unparsable} & \textbf{Ill-formed} & 3 \ 1.5 \ \\
& \textbf{Failure} & 19 \ 9 \ \\
\hline
\textbf{Total} & & 212 \ \\
\hline
\end{tabular}
\end{center}
\caption{Testing results}
\end{table}

The total number of sentences across the document was 212 sentences. The average sentence length was 10 words. The longest sentence was 44 words long. The result shows that the number of sentences parsed successfully was 146 sentences, about 68.8%. 44 sentences were assigned incorrect parse trees, about 20.7%. The number of sentences that were not parsed (has not produced a parse tree) was 22 sentences, about 10.3%.

The 44 sentences which produced grammatically incorrect parse trees, we do not consider the correctness of the parse fragments of the parse tree, which represent correct syntactic structure of parts of a sentence. In other words, the complete sentence is rejected if any of its parse fragments would be grammatically incorrect. Also, it is possible to decrease the total number of unparsable sentences by acquiring additional grammar rules.

Although the results observed in our experiment are satisfactory, the practical question is how many final parse trees we can get. This structural ambiguity may lead to an enormous amount of possible readings for an input sentence. Techniques will have to be developed to be able to deal with large numbers of analyses, and to be able to choose the most appropriate reading from such a set of candidate analyses. However, this is not covered in the current work.

Table 2 shows the ambiguities of sentences parsed successfully across the document. As shown, the maximum number of parse trees was limited to four parse trees. The total disambiguated sentences were 152. The total ambiguated sentences were 38.
Table 2 the relation between the number of trees and the number of sentences.

<table>
<thead>
<tr>
<th>Number of Trees</th>
<th>Number of sentences</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>152</td>
</tr>
<tr>
<td>2</td>
<td>28</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>

5. Conclusions
This paper has been concentrated on issues in the design and implementation of an Arabic Parser. The parser encodes the Arabic grammar rules of irab (‘إْيْأَرْأَب’ and the effects of applying these rules to the constituents of sentences. The system can be geared towards any other related system or application since it has been built as a module. The parser tries to avoid some problems associated with ambiguities. The extent to which ambiguity could be resolved is not comprehensive since ambiguity can be semantic which is another research problem. Experiment on real extension document was performed. The results observed in our experiment are satisfactory. For future work, the parser can be integrated into some other Arabic applications such as machine translation systems, systems for teaching Arabic, and system for checking and correcting grammatical errors.

References
Khayat M. Understanding Natural Arabic, in proceeding of the First KFUM workshop on information and computer science, Dhahran, Saudi Arabic, pp. k1-k4, 1996.
Shieber, S. M., Introduction to Unification-Based Approaches to Grammar, Center for the study of language and information, Stanford, 1986.