Int J Speech Technol (2016) 19:313–323
DOI 10.1007/s10772-015-9300-x

Parsing Arabic using induced probabilistic context free grammar

Nabil Khoufi[1] · **Chafik Aloulou**[1] · **Lamia Hadrich Belguith**[1]

Received: 4 May 2015 / Accepted: 12 August 2015 / Published online: 4 September 2015

Abstract The importance of the parsing task for NLP applications is well understood. However developing parsers remains difficult because of the complexity of the Arabic language. Most parsers are based on syntactic grammars that describe the syntactic structures of a language. The development of these grammars is laborious and time consuming. In this paper we present our method for building an Arabic parser based on an induced grammar, PCFG grammar. We first induce the PCFG grammar from an Arabic Treebank. Then, we implement the parser that assigns syntactic structure to each input sentence. The parser is tested on sentences extracted from the treebank (1650 sentences).We calculate the precision, recall and f-measure. Our experimental results showed the efficiency of the proposed parser for parsing modern standard Arabic sentences (Precision: 83.59 %, Recall: 82.98 % and F-measure: 83.23 %).

Keywords Parsing · Arabic language · PCFG grammar · Arabic treebank · Linguistic resource

✉ Nabil Khoufi
nabil.khoufi@fsegs.rnu.tn

Chafik Aloulou
chafik.aloulou@fsegs.rnu.tn

Lamia Hadrich Belguith
l.belguith@fsegs.rnu.tn

[1] ANLP-RG, MIR@CL Lab, University of Sfax, Sfax, Tunisia

1 Introduction

Parsing Arabic texts is not an easy task to perform because of two fundamental phenomena. The first phenomenon is related to the particularities of the Arabic language that make it more ambiguous than other natural languages. These characteristics influence its different levels of processing: morphological, syntactic, and semantic. The second phenomenon concerns the significant scarcity of available digital resources for the Arabic language, especially grammars and corpora.

Several studies have been conducted in order to solve issues related to parsing and to enhance parsers' performance. These efforts can be classified into three distinct approaches: the linguistic approach (symbolic), the numerical (or statistical) approach, and the mixed or hybrid approach. The linguistic approach uses lexical knowledge and language rules in order to parse sentences whereas numerical approaches are essentially based on statistics or on probabilistic models. This type of approach is mainly based on frequencies of occurrence that are automatically calculated from the corpora. The third approach is a hybrid approach that integrates both the linguistic and the numerical analysis (Khoufi et al. 2013).

In this paper we focus on the parsing task using the linguistic approach. This approach requires, in addition to the input sentence, some linguistic resources to guide the syntactic analysis. One method to provide such resources to the parser consists of writing down the language grammar manually. However, manual construction of such linguistic resources is a difficult task to undertake, and is time consuming. Unlike a programming language, natural language is far too complex to simply list all the syntactic rules. In addition, it is difficult to exhaustively list lexical properties

of words, and lastly, the written grammar has to be validated by Arabic linguists.

A second method to build linguistic resources is the use of treebanks as source of knowledge. Indeed, treebanks, as rich corpora with annotations, provide an easy way to build other linguistic resources, such as extensional and intentional lexicons, syntactic grammars, bilingual dictionaries, etc. This promotes their reuse and makes their implicit information explicit. Another advantage of treebanks is that they are not only developed and validated by linguists, but also submitted to consensus, which promotes their reliability. The possession of such a resource makes it possible to generate new resources based on other formalisms with wide coverage automatically in a very controlled manner. These resources inherit the original treebank qualities, while improving construction time.

This paper is organized as follows: Sect. 2 is devoted to presenting the Arabic language ambiguity. Sect. 3 gives an overview of works related to Arabic parsing using the symbolic approach. Sect. 4 gives PCFG basic definitions. Section 5 explains our method for parsing Arabic language and presents the induced grammar and experimental results. Section 6 provides the conclusion and perspectives.

2 Arabic language ambiguity

The Arabic language has specific characteristics that make it more difficult to parse than other natural languages. Besides classical phenomena like coordination, anaphora and ellipsis which exist in the Latin languages, there are other features specific to Arabic that generate problems in the parsing task.

The first one is the unvocalization phenomenon that gives rise to grammatical ambiguities. Indeed, graphic representations of words without vowels are not useful for disambiguating grammatical interpretations and semantic meanings. In fact, a word can have more than one grammatical interpretation. Consequently, unvocalized texts are more ambiguous than vocalized ones. According to Debili's statistics (Debili et al. 2001) 74 % of Arabic words accept more than one vocalization. Debili's statistics show that the grammatical ambiguity rate reaches 5.6 on average for vocalized words and 8.7 on average for unvocalized ones. Table 1 presents an example of an unvocalized word with its different vocalized forms.

Agglutination in Arabic is another specific phenomenon where articles, prepositions, pronouns, etc. can be affixed to adjectives, nouns, verbs and particles to which they are related. This phenomenon increases syntactic difficulties since it leads to exceptional structures. An agglutinative form can constitute a whole sentence, as in *wastaqbalahum* واستقبلهم (Then he welcomed them). Therefore, it requires some specific processing to find their correct syntactic structure.

Word order in Arabic is relatively free. Generally, we put the word that we want to focus on at the beginning of the sentence and we put the longest or the richest one (in meaning or tone) at the end. This free order leads to artificial syntactic ambiguities and complicates grammar construction. In fact, grammar rules should provide all possible combinations to describe all correct word orders in the sentence. Table 2 illustrates an example showing the effect of order change.

We can change the order of words in this sentence and obtain the two structures presented in Table 3 and 4.

Abundant use of recursive structures is another specificity of Arabic texts. Embedded structures are common in Arabic texts as well as in other natural languages.

However, it is more frequent in Arabic since some propositions can play a role in other propositions. Let us consider the following example:

الشرطة هي التي قبضت على المجرم الذي ضل هارباً مدة طويلة.

(The police have arrested the criminal who remained on the run for a long time).

It is a nominal sentence, while the proposition (خبر) is also a nominal sentence:

هي التي قبضت على المجرم الذي ضل هارباً مدة طويلة

(have arrested the criminal who remained on the run for a long time).

In this same example, even segmentation into sentences is not possible since there are many propositions that are not independent and do not belong to the same syntactic level. As a result, the lengths of Arabic sentences are not limited.

Table 1 Example of ambiguity due to the unvocalization phenomenon

Unvocalized word	Vocalized forms	Buckwalter transliteration	Translation
فهم	فَهِمَ	Fahima	He understood
	فَهَّمَ	Fah ~ ama	He explained
	فُهِمَ	Fuhima	It has been understood
	فَهْمٌ	Fahomn	Comprehension
	فَهُمْ	Fahumo	Then them
	فَهَمَّ	Faham ~ a	Then started
	…	…	…

Table 2 Example of an Arabic sentence, order 1

Sentence	إلى المدرسة.	الولد	ذهب
Gloss	To school	The boy	Went
Form	Complement	Subject	Verb

Table 3 Arabic sentence, order 2

Sentence	إلى المدرسة.	ذهب	الولد
Gloss	To school	Went	The boy
Form	Complement	Verb	Subject

Table 4 Arabic sentence, order 3

Sentence	الولد.	ذهب	إلى المدرسة
Gloss	The boy	Went	To school
Form	Subject	Verb	Complement

3 Related works

This work is part of a hybrid method for parsing Arabic language. This hybrid method (symbolic/statistical) aims at the collaboration of two parsers, the first one based on a statistical model obtained using supervised learning techniques(Khoufi et al. 2014) and the second based on the induced grammar described in this paper. Therefore in this section we focus on the presentation of parsers that are based on the use of a symbolic grammar.

Numerous studies are actively being conducted for this purpose. However, their number is very limited when compared to works dealing with other natural languages such as English, Spanish or French. To our knowledge, the majority of works regarding Arabic language parsing use the linguistic approach that yields satisfactory results, but does not attain the English state-of-the-art level yet.

Ouersighni (2001) developed a morpho-syntactic analyzer in modular form for Arabic. The analysis is based on the grammatical AGFL (Affix Grammars over a Finite Lattice) formalism. This parser generates clitics, prefixes, roots, and suffixes for each analyzed word in addition to its lexical forms, then constructs the whole syntactic tree of the sentence.

The analyzer of (Othman et al. 2003) also developed in a modular form is based on a grammar following the UBG (Unification Based Grammar) formalism. The constructed grammar is composed of 170 rules which mainly represent components' roles (subject, object, etc.) in a given sentence. Constraints are associated with UBG rules to control the quality of the obtained syntactic trees. The Othman parser proceeds with a Top-down strategy to parse sentences.

Aloulou (2005) had developed a parsing system called MASPAR (Multi-Agent System for Parsing Arabic) based on a multi-agent approach. MASPAR parses sentences by dividing tasks between six agents: a tokenization agent, a lexical agent, a morphological agent, a syntax agent, an anaphora agent, and an ellipsis agent. All these agents work together to parse input text. The author chose to use the Head-driven phrase structure grammar (HPSG) formalism arguing that it is a representation that minimizes the number of syntactic rules and provides rich and well-structured lexical representations.

McCord and Cavalli-Sforza (2007) developed a slot grammar (SG) parser for Arabic (ASG) with new features of SG designed to accommodate Arabic as well as the European languages for which SGs have been built. Slot Grammar is dependency oriented, and has the feature that allows both deep structure (via logical predicate arguments) and surface structure to be shown in parse trees. The authors focused on the integration of BAMA (Buckwalter's Arabic Morphological Analyzer) (Buckwalter, 2004) with ASG, and on a new, expressive SG grammar formalism (SGF) and they illustrated the way SGF is used to advantage in ASG.

The analyzer of Bataineh and Bataineh (2009) uses recursive transition networks to build a context free grammar which describes the most common sentences in Arabic. The transition network considers syntax rules as graphs, arcs and labels. These are finite state automata representing rules' transcripts. To represent the maximum of structures, a set of sentences' patterns was also derived from school texts. These patterns are converted to context free rules with the help of Arab linguists. A sentence is accepted by the grammar if it is generated by a complete course (without interruption) of these transitions' networks.

Klein and Manning (2003) developed a parser that implements a factored product model, with separate PCFG phrase structure and lexical dependency experts, whose preferences are combined by efficient exact inference, using an A* algorithm. As well as providing an English parser, the parser has been adapted to work with other languages. A Chinese parser based on the Chinese Treebank, a German parser based on the Negra corpus and Arabic parsers based on the Penn Arabic Treebank are also included (Green and Manning 2010).

Al-Taani et al. (2012) constructed a grammar under the CFG formalism (Context Free Grammar) then implemented it in a parser with a top-down analysis strategy. This parser focused on identifying sentence type (Nominal or verbal) and domain words.

Table 5 Comparative study of Arabic parsers

Authors	Grammar formalism	Parsing strategy	Testing data	Results
Ouersighni (2001)	AFGL grammar	RBP[a] strategy	105 sentences	Precision 76.19 %
Othman et al. (2003)	UBL grammar	–	–	–
Aloulou (2005)	HPSG grammar	Multi agents strategy	3871 sentences (<11 words)	61 % correct 16.5 % partly correct 22.5 % incorrect
McCord and Cavalli-Sforza (2007)	ASG grammar	Bottom-up	1000 sentences (13–20 words)	72 % complete parses (with no guaranty of correctness)
Bataineh and Bataineh (2009)	CFG grammar	Top-down	90 sentences	85.4 % correct 2.2 % incorrect 12.4 % rejected
Green and Manning (2010) (Stanford parser)	Human interpretable grammars (PCFG based)	–	ATB 10 %	Precision 81.07 % Recall 80.27 % F-measure 80.67 %
Al-Taani et al. (2012)	CFG grammar	Top-down	70 sentences (2–6 words)	Precision 94 %
Alqrainy et al. (2012)	CFG grammar	Top-down	105 sentences	Accuracy 95 %.

[a] Recursive backup parser

The work of (Alqrainy et al. 2012) presents a simple parser for Arabic sentences. The aim of this parser was to check whether the syntax of an Arabic sentence is grammatically correct by constructing a new, efficient Context-Free Grammar. Alqrainy designed the parser to take advantage of the top-down technique. He used the NLTK (Natural Language ToolKit) tool (Bird et al. 2009) to build and test the Arabic CFG grammar.

Table 5 summarizes our comparative study of Arabic parsers. The comparison is performed using these criteria:

- Grammar formalism
- Parsing strategy,
- Size of the testing data,
- Results as precision, accuracy or error scores.

4 PCFG preliminary

A probabilistic context-free grammar (PCFG) also called stochastic CFG (SCFG), is an extension of the famous context-free grammar, where a certain probability is assigned to each rule. Probabilistic context-free grammars are defined by a 5-tuplet < N, T, R, S, P> as follows:

- N is a finite set of non-terminal symbols.
- T is a finite set of terminal symbols.
- R is a finite set of rules *ri* of the form $X \rightarrow Y1Y2\ldots Yn$, where $X \in N$, $n \geq 0$, and $Yi \in (N \cup T)$ *for* $i = 1 \ldots n$.
- S ∈ N is a distinguished start symbol.
- P is the set of probabilities *pi* associated to rules *ri* where: $\sum P(X \rightarrow Yi) = 1$, $\forall X \in N$ *and* $Yi \in (N \cup T)$ *for* $i = 1 \ldots n$.

Note that some sentences may have more than one underlying derivation in case of the use of a classic CFG and therefore generate several parse trees. Therefore probabilities P in a PCFG are used to produce the most likely parse tree for a given sentence. The probability of a parse tree is obtained by multiplying the probability of each rule used at each node of the tree.

In the following we give an example of a PCFG and two possible parse trees (Fig. 1) for a sentence to understand the parsing ambiguity and the solution offered by the PCFG to address it.

N = {S, VP, NP, ADJP, V, NN, DET + NN, CONJ, PRON, DET + ADJ}

S = S

T = {لعب, الطفل, و, ابن, جار, هم, المؤدب}

R, P=

S → VP NP	[1.0]	V → لعب	[1.0]
VP → V	[1.0]	NN → جار	[0.4]
NP → NP CONJ ADJP	[0.3]	NN → ابن	[0.6]
NP → NP CONJ NP	[0.1]	DT + NN → الطفل	[1.0]
NP → DET + NN	[0.3]	CONJ → و	[1.0]
NP → NN ADJP	[0.1]	PRON → هم	[1.0]
NP → NN PRON	[0.1]	ADJ + DET → المؤدب	[1.0]
NP → NN NP	[0.1]		
ADJP → NP DET + ADJ	[0.6]		
ADJP → NP DET + NN	[0.4]		

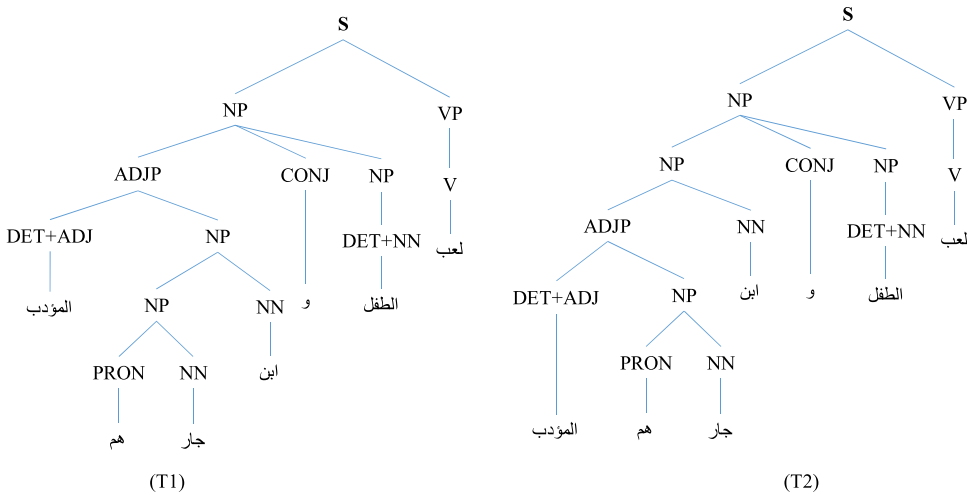

Fig. 1 Two possible parse trees (derivations) for the sentence: لعب الطفل وابن جارهم المؤدب

Arabic sentence:	لعب الطفل وابن جارهم المؤدب.
Buckwalter transliteration	*laEiba AlTiflu w ibnu jArihum Almu&ad ~ abu*

Two possible translations for the sentence:

لعب الطفل وابن جارهم المؤدب.

- The child played with his polite neighbor's son.
- The child played with his neighbor's polite son.

Figure 1 presents two possible parse trees (derivations) for the above mentioned sentence, both of which are valid under a classic CFG without consideration of the probabilities.

This example is a case of adjectival phrase attachment ambiguity: the adjective (ADJ) *Almu&ad ~ abu* "المؤدب" (polite) can modify either the neighbor's son *ibnu jArihum* (ابن جارهم), or the neighbor himself *jArihum* (جارهم). In the first parse tree shown in Fig. 1, the ADJ modifies the neighbor's son, which means that the neighbor's son is polite. In the second parse-tree (T2), the ADJ modifies the neighbor only, which in this case means that the neighbor is polite.

The probability of an entire tree is the product of probabilities for these individual choices. We multiply the P values of each PCFG rule that it contains thus obtaining:

$P(T1) = P(S \rightarrow VP\ NP) \times P(VP \rightarrow V) \times P(V \rightarrow$لعب$) \times P(NP \rightarrow NP\ CONJ\ ADJP) \times P(NP \rightarrow DET + NN) \times P(DET + NN \rightarrow$الطفل$) \times P(CONJ \rightarrow$ و$) \times P(ADJP \rightarrow NP\ ADJ) \times P(NP \rightarrow NN\ NP) \times P(NN \rightarrow$ابن$) \times P(NP \rightarrow NN\ PRON) \times P(NN \rightarrow$جار$) \times P(PRON \rightarrow$هم$) \times P(DET + ADJ \rightarrow$المؤدب$)$

$P(T1) = 1.0 \times 1.0 \times 1.0 \times 0.2 \times 0.3 \times 1.0 \times 1.0 \times 0.6 \times 0.1 \times 0.6 \times 0.1 \times 0.4 \times 1.0 \times 1.0 = 0.0000864$

$P(T2) = P(S \rightarrow VP\ NP) \times P(VP \rightarrow V) \times P(V \rightarrow$لعب$) \times P(NP \rightarrow P\ CONJ\ ADJP) \times P(NP \rightarrow DET + NN) \times P(DET + NN \rightarrow$الطفل$) \times P(CONJ \rightarrow$ و$) \times P(NP \rightarrow NN\ ADJP) \times P(NN \rightarrow$ابن$) \times P(ADJP \rightarrow NP\ DET + ADJ) \times P(NP \rightarrow NN\ PRON) \times P(NN \rightarrow$جار$) \times P(PRON \rightarrow$هم$) \times P(DET + ADJ \rightarrow$المؤدب$)$

$P(T2) = 1.0 \times 1.0 \times 1.0 \times 0.2 \times 0.3 \times 1.0 \times 1.0 \times 0.2 \times 0.6 \times 0.6 \times 0.1 \times 0.4 \times 1.0 \times 1.0 = 0.0001728$

In the following section, we present our method for parsing Arabic using an induced PCFG.

5 Our method

Our method for parsing Arabic language has two phases: the grammar induction phase and the parsing phase. The first phase uses an annotated corpus as a knowledge source for the induction of the PCFG, whereas the second phase implements the induced grammar (results of the first phase) to achieve parsing. The phases of our approach are illustrated in Fig. 2.

The two phases of the method are described in the following subsections.

5.1 PCFG grammar induction from the PATB Treebank

Our objective in this phase is to automatically induce a PCFG from an annotated corpus. This process consists of two steps: The first step is to induce CFG rules from the annotated corpus. The second step is to assign a probability to each induced rule. The application of these two steps allows us to obtain a PCFG. Figure 2 illustrates the workflow of this first phase.

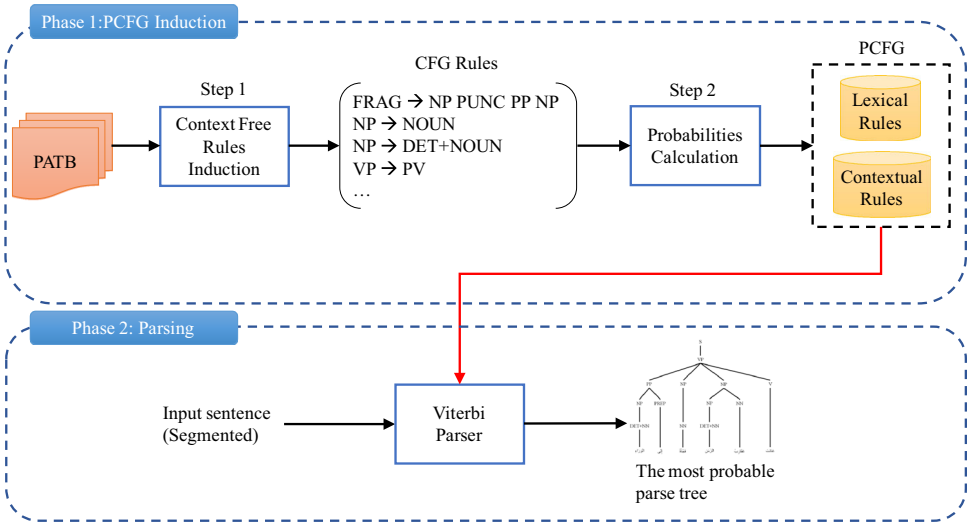

Fig. 2 Architecture of the proposed method

Since our construction of the PCFG grammar is based on an annotated corpus, we begin by discussing the corpus we used; then we describe the PCFG induction process in detail.

5.1.1 Using the Penn Arabic Treebank

In our work, we chose to use the well-known corpus, the Penn Arabic Treebank (PATB). This choice was motivated not only by the richness, the reliability and professionalism with which it was developed but also by the syntactic relevance of its source documents (converted to several other Treebank representations). Indeed, its annotations have the advantage of being reliable. This is shown by its efficacy in a large number of research projects in various fields of NLP (Habash, 2010). The good quality of the text and its annotations is demonstrated by its performance in the creation of other Arabic Treebanks such as the Prague Arabic Dependency Grammar (Hajic et al. 2001) and the Columbia Arabic Treebank (Habash and Roth, 2009), which converted the PATB to its syntactic representations in addition to other annotated texts.

Indeed, these annotations were manually elaborated and validated by linguists. Moreover, this treebank is composed of data from linguistic sources written in Modern Standard Arabic. This corpus is also the largest Arabic corpus which integrates syntactic tree files. The use of a large amount of annotated data in a grammar construction process increases the quality of the generated linguistic resource.

The PATB was developed in the Linguistic Data Consortium at the University of Pennsylvania (Maamouri et al. 2004). Texts in the corpus, as with most texts written for adults in Modern Standard Arabic, such as newspaper articles, contain no vowels.

We used the PATB 3 version 3.2 of this corpus Maamouri et al. 2010), which consists of 599 files, and includes POS tags, morpho-syntactic structures at many levels and glosses. It comprises 402 291 tokens and 12 624 sentences. It is available in various formats: The "sgm" format refers to source documents. The "pos" format gives information about each token as fields before and after clitic separation. The "xml" format contains the "tree token" annotation after clitic separation. The "penntree" format generates a Penn Treebanking style. And finally the "integrated" format brings together information about the source tokens, tree tokens, and the mapping between them and the tree structure.

5.1.2 Description of the induction process

As shown in Fig. 2, the first step of our method is the induction of CFG rules, including duplicates, which will be used in the second step. A deep study of the PATB allows us to identify the rules that guide the CFG rules induction process. We focused on the morpho-syntactic trees of the PATB and we identified the following rules:

R1 *Tree root → Start symbol*
R2 *Internal tree node → Non terminal symbol*
R3 *Tree word → Terminal symbol*
R4 *Tree fragment → CFG rules*

We noticed that each parse tree is a sequence of context-free rules and each one has the same symbol "S" at its root. Thus, the root symbol "S" is taken to be the start symbol (S) of the grammar. Non-terminal (N) symbols consist of the set of internal nodes of the whole parse tree. The set of all words seen in the trees (the leaves) compose the set (T) of terminal symbols. Edges between the nodes of the trees are used to induce CFG rules(R). Figure 3 presents the

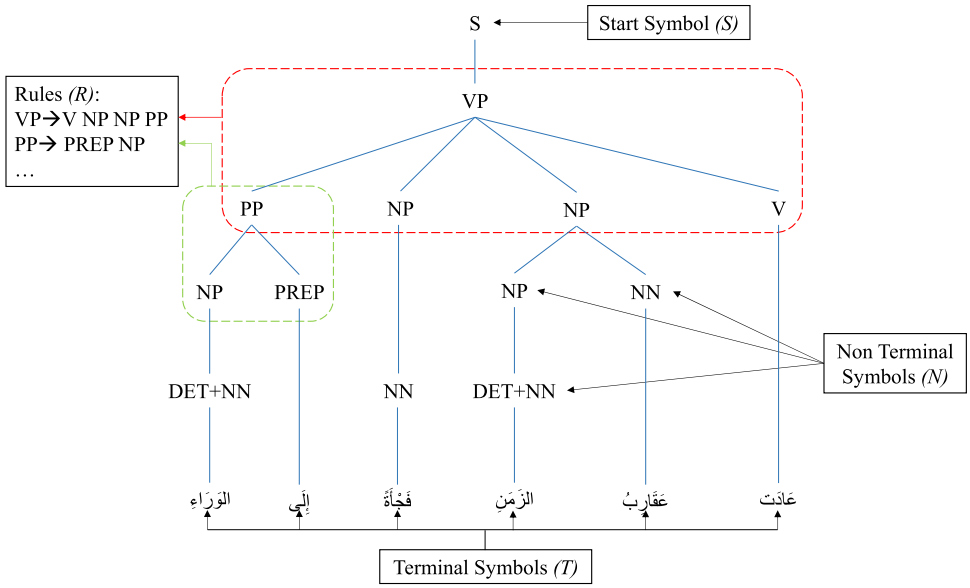

Fig. 3 Induction of elements of rules from a parse tree fragment

process of induction of PCFG elements (S, N, T, R) from a PATB parse tree.

Note that tags on the right hand-side of the induced rules are in reversed order compared to the parse tree. This is due to the reading orientation of the Arabic language which is from right to left.

In Arabic, the word and its determinant are agglutinated as seen in the word *alzaman* الزَمَنِ (Noun). We chose to keep these elements together in one rule to reduce grammar size. This choice does not influence the analysis quality since there is no loss of information.

Noun → Det Noun
Det → ال
Noun→زمن

⟹

Noun → Det+Noun
Det+Noun → الزمن

After application of the first step of our method to the example of Fig. 3, we obtain 12 CFG rules composed of 6 contextual rules and 6 lexical rules as presented in Table 6.

Once CFG rules had been induced from the PATB with all duplicates, we moved to the second step consisting of the calculation of rule probability (P) to finally obtain the PCFG grammar. Each rule probability is estimated using the following formula:

$$P(X \rightarrow Y) = \frac{Count(X \rightarrow Y)}{Count(X)}$$

where Count (X → Y) is the number of times the rule X → Y is seen in the Treebank and Count (X) is the count of rules that have the non-terminal X on the left-hand side. For example, the rule VP → V NP PP is seen109 times in the PATB and we have counted 1133 rules that have VP on the left-hand side, thus:

$$P(VP \rightarrow V\ NP\ PP) = \frac{109}{1133}$$

Table 6 CFG Rules induced from the example of Fig. 3

Contextual rules	Lexical rules
S → VP	V → عَادَت (turned)
VP → V NP NP PP	NN → عَقَارِبُ (the hands)
NP → NN NP	DET + NN → الزَمَنِ (time)
NP → DET + NN	NN →فَجْأَةً (suddenly)
NP → NN	PREP → إِلَى (to)
PP → PREP NP	DET + NN → الوَرَاءِ (backwards)

In the following section, we describe our experiment and explain some interesting information about the grammar we obtained.

5.1.3 Grammar induced from the PATB

After applying our method, we obtained the PCFG grammar. As we mentioned earlier, The PATB is a very rich corpus and it contains many annotations like mood, gender, number, etc. The PATB corpus is annotated using a large set of annotations, which gives a high level of granularity. For example, the Part of speech annotation tag set contains 498 tags that provide much information like gender, mood, etc. (Maamouri et al. 2008). There are also 22 syntactic category tags and 20 tags that describe semantic relations between tokens. In addition, stop words, which are very numerous in the Arabic language, are also annotated with specific tags.

Table 7 Most frequent LHS rules

Left-hand symbol (LHS)	NP	VP	S	FRAG	ADJP	UCP	PP
Rule count	1611	1133	1013	307	243	155	118

Table 8 Rule count

Contextual rules	4661
Lexical rules	38,901
Total	43,562

Table 9 Evaluation results

Precision	Recall	F-measure
83.59 %	82.98 %	83.23 %

Incorporation of all this information within the grammar increases its complexity and its size. Its size depends on the granularity level of the categories it describes. The higher this level, the more complex these grammars are, but also the more respectful of language specificity. For example, this tag set {ADJ + NSUFF_FEM_DU_NOM, ADJ + CASE_DEF_ACC, ADJ + CASE_DEF_GEN, ADJ + CASE_DEF_NOM} describes adjectives with a high level of granularity. If we reduce the granularity level to a minimum, these tags will be reduced into one tag, ADJ. This reduction influences the number of grammar rules. We chose to reduce the POS tags to the basic tags, which leaves about 70 tags for the Arabic language, to facilitate the use of induced grammars for NLP applications.

There are also other tags in the PATB that are generated during the initial tagging and parsing process like *ICH*, *O*, *RNR*, NONE and NAC. Indeed those tags, if considered in a parsing task, will increase the number of rejected parses because they describe morphologic tagging errors and empty categories. Therefore, these tags were removed from our tag set to make our grammar more consistent.

We present below some statistics about our PCFG grammar. Table 7 presents the most frequent PCFG syntactic rules generated from the PATB corpus after applying our method and Table 8 presents the overall count of rules (contextual rules and lexical rules).

5.2 Parsing experiments

In order to evaluate the performance of the induced grammar in the parsing task, we divided the PATB corpus following the "Mona Diab" or "Johns Hopkins 2005 Workshop" splits. We used the major part for induction of the PCFG grammar (90 %) and the second part for parsing tests (10 %).

We used Viterbi implementation of the NLTK tool (Bird et al. 2009) to test PCFG in the parsing task. Indeed, several algorithms can be used in parsing using a PCFG; the most famous ones are CYK (the Cocke-Younger-Kasami algorithm), Earley and Viterbi. This choice is justified by:

1. Viterbi can be used with any probabilistic context free grammar but CYK needs to use a grammar in Chomsky Normal Form. This transformation increases grammar size and therefore parsing time.
2. Viterbi parses in linear time thus it is less complex than Earley and CYK which both have cubic worst case time.

Viterbi is a bottom-up PCFG parser that uses dynamic programming to find the single most likely parse for an input text. The Viterbi parser parses texts by filling in a "most likely constituent table".

In order to find the most likely constituent with a given span and node value, the "Viterbi Parser" considers all productions that could produce that node value. For each production, it finds all children that collectively cover the span and have the node values specified by the production's right hand side. If the probability of the tree formed by applying productions to the children is greater than the probability of the current entry in the table, then the table is updated with this new tree.

5.2.1 Parsing results

In order to estimate the performance of our parser, we calculated the Precision, Recall and F-measure for each tested sentence using the following formulas:

- Precision = cardinality (Reference ∩ Test)/cardinality (Test)
- Recall = cardinality (Reference ∩ Test)/cardinality (Reference)
- F-measure = 2 × ((Precision × Recall)/(Precision + Recall))

Then we calculate the macro average of the obtained values. The test set is composed of 1650 sentences extracted from the PATB. The average length of the sentences is about 21words (the length ranges between 3 and 40 words). Obtained results are exposed in the Table 9.

The evolution of the precision, recall and f-measure while increasing the length of the parsed sentence can be seen in Fig. 4.

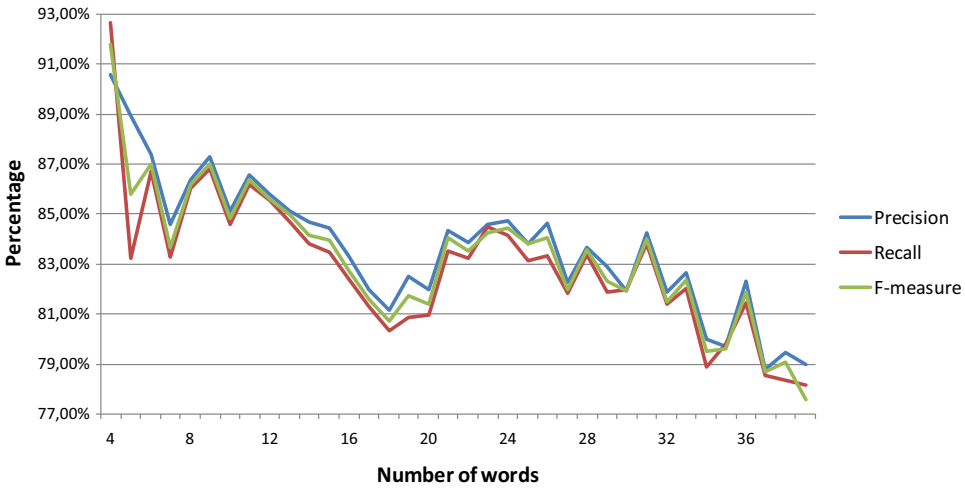

Fig. 4 Precision, recall and F-measure progression

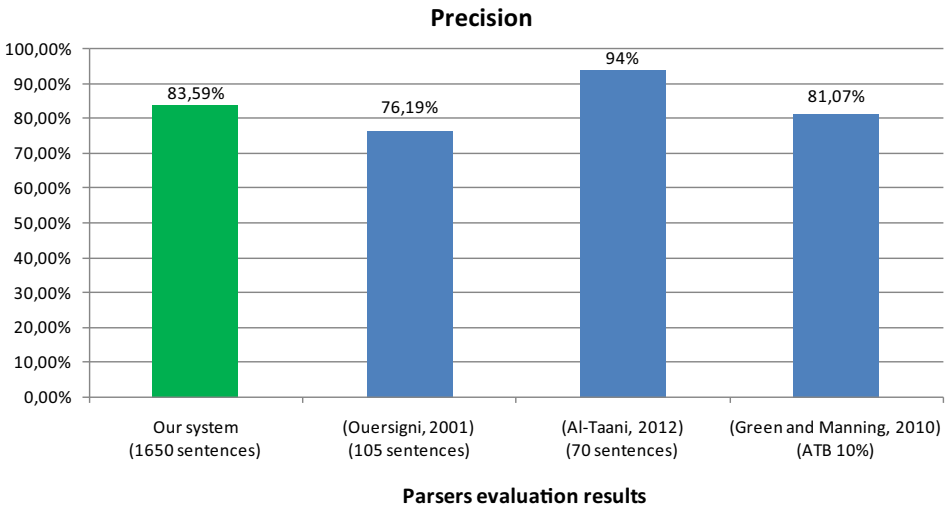

Fig. 5 Precision comparisons

Our results are interesting considering the state-of-the-art for Arabic parsing performance. Nevertheless, it is difficult to compare our results to the ones from existing Arabic parsers because different evaluation metrics and different test sets were used. Moreover, state-of-the-art systems are not available online and the testing data are not compatible, which complicates performance comparison.

Our Evaluation has two highlights which are:

- The size of the testing data (1650 sentences) which is widely superior to other works' testing data (see Table 5).
- The length of the sentence which is between 3 and 40 words with an average length of 21.16 words.

Figure 5 presents a comparison between our parser and three state-of-the-art parsers that calculate the same evaluation metrics.

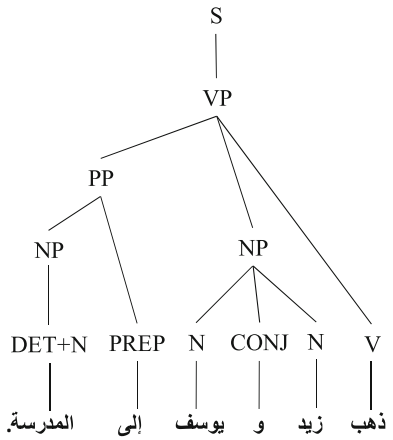

Fig. 6 Parsed sentence from Al-Taani testing data (parse tree)

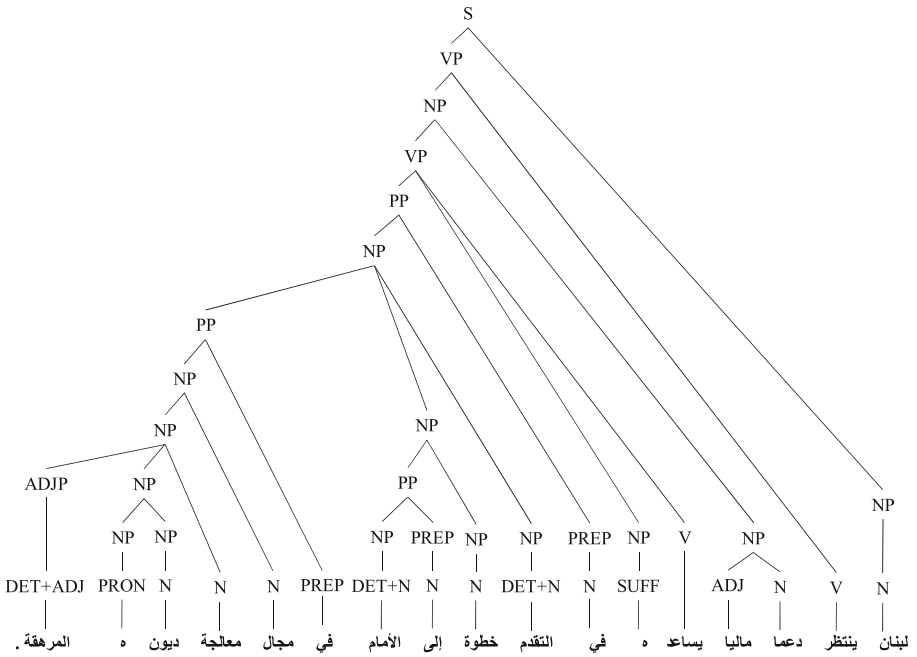

Fig. 7 Parsed sentence from our testing data (parse tree)

(Lebanon is waiting for financial support to help in progress step forward in addressing its burdensome debts.)

Figure 5 highlights the high level of precision attained by our method, which noticeably outperforms Ouersighni (2001) and is still very competitive with Al-Taani et al. (2012). Our results are also slightly better (about 2.52 %) than the Arabic version of Stanford parser (Green and Manning 2010) knowing that we use the same PATB slits.

We think that this difference might be due to the size of the testing data, the length of the tested sentences and the level of complexity. Indeed, our testing data (1650 sentences) impressively outnumbers Al-Taani's testing data (70 sentences) in addition to the fact that our testing sentences are longer (average length 21 words) than those of Al-Taani (average length 3.93 words) which complicates the parsing task. To illustrate the complexity of our tested sentences, we present in Fig. 6 and 7 two parsed sentences, one from the Al-taani testing data and the second from our testing data, respectively.

6 Conclusion and perspectives

We presented in this paper our method for parsing the Arabic language based on an Arabic PCFG (Probabilistic context free grammar) grammar. Our method consists of two phases: in the first one we induced a PCFG grammar from the PATB parse trees using induction rules. In the second we used the induced grammar and the Viterbi parsing algorithm to parse Arabic sentences. We tested our parser on sentences extracted from the PATB (over 1650 sentences) and we achieved encouraging and very satisfactory results (precision: 83.59 %, recall: 82.99 % and f-measure: 83.24 %). As a future work, we plan to integrate a morphological analyser sush as MADAMIRA (Pasha et al. 2014) to cover words that are not covered by lexical rules, then update the database of lexical rules. Our long term goal is to integrate this work into a hybrid method for parsing Arabic. This hybrid method would allow collaboration of two parsers, the first is based on a statistical model obtained using supervised learning techniques (Khoufi et al. 2014) and the second is based on the induced grammar described in this paper.

References

Aloulou, C. (2005). Une approche multi-agent pour l'analyse de l'arabe : Modélisation de la syntaxe. Doctoral dissertation, University of Manouba, Tunisia

Alqrainy, S., Muaidi, H., & Alkoffash, M. S. (2012). Context-free grammar analysis for Arabic sentences. *International Journal of Computer Applications, 53*(3), 7–11.

Al-Taani, A., Msallam, M., & Wedian, S. (2012). A top-down chart parser for analyzing Arabic sentences. *The International Arab Journal of Information Technology, 9*, 109–116.

Bataineh, B. M., & Bataineh, E. A. (2009). An efficient recursive transition network parser for Arabic language. In *Proceedings of the World Congress on Engineering*, vol 2 (pp. 1–3)

Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with python*. Sebastopol: O'Reilly Media Inc.

Buckwalter T. (2004). 'Buckwalter Arabic morphological analyzer version 2.0′.

Debili, F., Achour, H., & Souissi, E. (2001). La langue Arabe et l'ordinateur: De l'etiquetage grammatical à la voyellation automatique, Correspondances 71 (1), Lyon, (pp. 1–20).

Green, S., and Manning, C. D. (2010). Better Arabic parsing: Baselines, evaluations, and analysis. In *Proceedings of the 23rd international conference on computational linguistics* (pp. 394–402). Baltimore: Association for Computational Linguistics.

Habash, N. Y. (2010). Introduction to Arabic Natural Language Processing. Synthesis Lectures on Human Language Technologies, G. Hirst, (Series Ed). 3(1).

Habash, N. Y., & Roth, R. M. (2009). Catib: The Columbia Arabic Treebank. In *Proceedings of the ACL-IJCNLP 2009 conference short papers* (pp. 221–224). Stroudsburg, PA: Association for Computational Linguistics.

Hajic, J., Vidová-Hladká, B., & Pajas, P. (2001). The Prague dependency treebank: Annotation structure and support. In *Proceedings of the IRCS workshop on linguistic databases* (pp. 105–114).

Khoufi, N., Aloulou, C., & Hadrich Belguith, L. (2014) Chunking Arabic texts using conditional random fields, In *Proceedings of the 11th ACS/IEEE international conference on computer systems and applications (AICCSA 2014)* (pp. 428–432), November 2014, Doha.

Khoufi, N., Louati, S., Aloulou, C., & Hadrich Belguith, L.(2013) Supervised learning model for parsing Arabic language, In *Proceedings of the 10th International workshop on natural language processing and cognitive science* (NLPCS 2013) (pp. 129–136), Marseille.

Klein, D., & Manning, C. D. (2003). Fast exact inference with a factored model for natural language parsing. *Advances in Neural Information Processing Systems 15 (NIPS 2002), Cambridge* (pp. 3–10). MA: MIT Press.

Maamouri, M., Bies, A., Buckwalter, T., & Mekki, W. (2004). The Penn Arabic Treebank: Building a large-scale annotated Arabic corpus. The NEMLAR conference on Arabic language resources and tools, pp. 102–109.

Maamouri, M., Bies, A. and Kulick, S. (2008). Enhancing the Arabic Treebank: A collaborative effort toward new annotation guidelines. In *Proceedings of the sixth international conference on language resources and evaluation* (LREC 2008), Marrakech May 28-30, 2008.

Maamouri M., Bies A., Kulick S., Krouna S., Gaddeche F. & Zaghouani W. (2010). Arabic Treebank: Part 3 v 3.2 LDC2010T08. Web Download. Philadelphia: Linguistic Data Consortium.

McCord, M. C., & Cavalli-Sforza, V. (2007). An arabic slot grammar parser. In *Proceedings of the 2007 Workshop on computational approaches to semitic languages: Common issues and resources* (pp. 81–88). Baltimore: Association for Computational Linguistics.

Othman, E., Shaalan, K., and Rafea, A. (2003). A chart parser for analyzing Modern Standard Arabic sentences. In *Proceedings of the MT summit IX workshop on machine translation for semitic languages: issues and approaches* (pp. 37–44).

Ouersighni, R. (2001). A major offshoot of the DIINAR-MBC project: AraParse, a morphosyntactic analyzer for unvowelled Arabic texts. *ACL 39th Annual Meeting, Stroudsburg* (pp. 9–16). Association for Computational Linguistics: PA.

Pasha, A., Al-Badrashiny, M., Diab, M., El Kholy, A., Eskander, R., Habash, N., Pooleery M., Rambow O., & Roth, R. M. (2014). Madamira: A fast, comprehensive tool for morphological analysis and disambiguation of arabic. In *Proceedings of the language resources and evaluation conference (LREC)*, Reykjavik.