A property grammar-based method to enrich the Arabic treebank ATB

Raja Bensalem, Kais Haddar, Philippe Blache

A property grammar-based method to enrich the Arabic treebank ATB

Raja Bensalem Bahloul[1], Kais Haddar[1], Philippe Blache[2]

[1]Multimedia InfoRmation systems and Advanced Computing Laboratory, Higher Institute of Computer Science and Multimedia, Sfax, Tunisia
raja_ben_salem@yahoo.com; kais.haddar@yahoo.fr
[2]Laboratoire Parole et Langage, CNRS, Université de Provence, France
pb@lpl.univ-aix.fr

Abstract. We present a method based on the formalism of Property Grammars to enrich the Arabic treebank ATB with syntactic constraints (so-called properties). The Property Grammar formalism is an effectively constraint-based approach that directly specifies the constraints on information categories. This can facilitate the enrichment process. The latter is based on three phases: the problem formalization, the Property Grammar induction from the ATB and the treebank regeneration with a new syntactic property-based representation. The enrichment of the ATB can make it more useful for many NLP applications such as the ambiguity resolution. This allows also the acquisition of new linguistic resources and the ease of the probabilistic parsing process. This enrichment process is purely automatic and independent from any language and source corpus formalism. This motivates its reuse. We obtained good and encouraging experiment results and various properties of different types.

Keywords: Arabic language, Property Grammar, treebank enrichment.

Introduction

The Property Grammar (GP) formalism [6] is a constraint-based approach that puts the constraint notion at the core of the linguistic analysis. In fact, it does not require the construction of a local structure of the syntactic information before using the constraints it described. Instead, it specifies directly the syntactic information on categories. The other approaches of the same family, however, have other directives. For example, the HPSG (Head-driven phrase structure grammar) needs a local tree [16] and the CDG (Constraint Dependency Grammar) a dependency relation [12]. Moreover, the GP formalism differs from other constraint-based approaches by its simple, direct, local and decentralized representation of linguistic information. Indeed, unlike generative theories, the GP represents, independently, all kinds of information, regardless their position, and even the partial, incomplete or non-canonical information. This promotes its flexibility and robustness. In addition, using the GP formalism can be favorable to parsing process. In effect, many implicit syntactic structures and relations become explicit thanks to this formalism. The specified qualities of the GP for-

malism encouraged us to use it in the development of a new resource enriched with properties for the Arabic language. Processing this language presents several challenges. These challenges are not only related to certain Arabic specificities to be studied (such as the lack of vowels and diacritic marks and the agglutinative aspect of words), but also to particular linguistic phenomena to be addressed (like the relatives, the anaphora and the coordination). The enrichment task is not easy and direct but requires verification modules of the GP properties in the treebank and matching functions of treated categories. The formalization phase is also challenging. It needs to choose an adequate model and to understand all of the treebank data to succeed the enrichment issue resolution.

The present paper fits in this context. Our goal is to describe the enrichment method of the Arabic treebank ATB with data acquired from a given GP. As a result, we obtain the first Arabic treebank enriched with varied syntactic properties available in variable granularity level according to the user needs. We may also specify the most relevant properties thanks to the frequencies of the treebank categories and properties. This may ease the probabilistic parsing process and evaluate the difficulty of processing cognitive systems. Moreover, new linguistic resources can be obtained from the enriched ATB such as syntactic lexicons and dependency grammars. The proposed enrichment method is based on three phases: the problem formalization, the GP induction from the source treebank ATB and the new treebank generation based on syntactic property. We stared with an empirical phase, a linguistic study of some Arabic sentence structures before testing the method on a large corpus (i.e. the treebank ATB). This task helps us to verify the correctness of the interpretations of the syntactic properties and their validity (satisfaction), and efficiency.

This paper is organized as follows: Section 2 is devoted to a brief presentation of related works. Section 3 proposes an Arabic linguistic study within the GP formalism. Section 4 describes the formalization phase. Section 5 explains our enrichment method. Section 6 shows the constraint solver descriptions. Section 7 presents experimental results and discussions. Section 8 gives a conclusion and some perspectives.

Related works

Before quoting some related works, it is necessary to present the main key concepts to use in our contribution: the GP and the ATB. The GP is based on a formalism [6] representing linguistic information through properties (constraints) in a local and decentralized manner. These properties express the relations that may exist between the categories composing the described syntactic structure. Syntactic properties in particular have six types: the linear order ($\prec$), the obligation of co-occurrence ($\Rightarrow$), the interdiction of co-occurrence ($\otimes$), the dependency ($\rightsquigarrow$), the interdiction of repetition (Unic) and the head (Oblig).

The ATB [11], is the richest Arabic treebank in reliable annotations (POS tags, syntactic and semantic hashtags), which are also compatible to consensus developed and validated by linguists. Its source documents are relevant, varied and large. They are even converted by several other treebanks into their representations. The ATB

grammar is adapted to the Modern Standard Arabic and has a phrase-based representation, which is consistent with the GP hierarchical structure.

Several works are proposed to enrich treebanks in different languages. The contribution of Müller [13] is an instance of converted treebanks. It proposes an annotation of Morphology and NP Structure in the Copenhagen Dependency Treebanks (CDT), which represents different parallel treebanks in many languages such as Danish, English, German, Italian, and Spanish.

The annotations to add in treebanks can be also organized according to well-defined linguistic formalisms. Thus, Oepen and al. [14] followed this directive by developing the Lingo Redwoods, which is a dynamic treebank. This new type of treebank parses analyzed sentences from ERG (English Resource Grammar) according to a precise HPSG formalism. The CCG formalism is another formalism chosen in the treebank enrichment methods. This is particularly the case of the contributions of Çakıcı [7], who created CCGbanks by converting the syntax graphs in the Turkish treebank into CCG derivation trees.

For French, Blache and Rauzy [6] proposed an automatic method, which hybridizes the constituency treebank FTB with constraint-based descriptions using the GP formalism. In addition, this method enriches the FTB with evaluation parameters of the sentence grammaticality.

For Arabic, which is the language that interests us the most, we can find some other works to enrich the ATB. They focus on improving this treebank with new richer annotations or on converting it into new formalisms. The OntoNotes project [10] and the Proposition Bank project (Propbank) for Arabic [15] are some instances of treebank extensions. The latter incorporate semantic level annotations. The contribution of Alkuhlani and Habash [2] provides an enrichment, which adds annotations that models attributes of the functional gender, number and rationality. The work of Abdul-Mageed and Diab [1] has even touched the sentimental level by associating specific annotations to the ATB sentences. There is also the work of Alkuhlani and al. [3], but it enriches the Columbia Arabic Treebank (CATiB) with the most complicated POS tags and lemmas applied in the ATB [11].

As regards the enrichment by employing new formalisms in the treebank source, we can refer to some examples that generates new treebanks: the Habash and Rambow contribution [9] with a TAG grammar, the Tounsi and al. contribution [17] with an LFG grammar and the El-taher et al. contribution [8] with a CCG grammar. Regarding the GP formalism, it was previously hybridized with the French treebank FTB as we have already mentioned.

By inspecting all the works cited above, we may figure out that none of them presents an in-depth formalization phase before proposing the enrichment approach. The absence of this phase can make the establishment of their approaches more difficult due to the lack of pre-specified needed data and the risk of having redundant treatments.

In addition, the enrichment of treebanks can be considered as a Constraint Satisfaction Problem (CSP). In this case, the ATB enrichment processes with new formalisms (TAG, LFG and CCG), which are mentioned above, will be tough. In fact, their representations would require a construction of local structures before referring to the constraints. As already mentioned in Section 1, the GP is an approach extremely based on constraint satisfaction. Its application in the ATB enrichment, we can solve

these limitations by directly accessing to the variable values of the problem through its categories. An Arabic GP in variable granularity is already available [5]. It is not manually built but automatically generated from an ATB part [4].

Arabic Linguistic study within the GP formalism

In order to evaluate the performance of the syntactic properties in the treebank, we followed a linguistic study of some examples of Arabic syntactic structures (phrases). In this study, we present different syntactic relations between these examples to explain the interpretation and the aspects related to each property type in the GP formalism. Therefore, we introduce six examples of Arabic sentences shown in Table 1 below:

S	فـي خارجهـا أيضـا.ظهر ذلك لـيس فـي تـونس وحـدهـا بـل	أدى ذلك إلى "تحالف دولي".
T	Zahar+a *'lika layosa fiy tuwnis+a waHod+a-hA, balo fiy xArij+i-hA >ayoD+AF .	->ad~aY *'lika <ilaY " taHAluf+K duwaliy~+K " .
G	This appeared not only in Tunisia but also abroad.	This led to an "international coalition".
P	(S (VP (PV Zahar+a) (NP (PRON *'lika)) (PP (PRT (PART layosa)) (PREP fiy) (NP (NP (NOUN tuwnis+a)) (ADJP (ADJ waHod+a-) (NP (PRON -hA))))) (PUNC ,) (CONJ balo) (PP (PREP fiy) (NP (NOUN xArij+i-) (NP (PRON -hA)))) (ADVP (ADV >ayoD+AF)))) (PUNC .))	(S (VP (PV ->ad~aY) (NP (PRON *'lika)) (PP (PREP <ilaY) (PUNC ") (NP (NOUN taHAluf+K) (ADJ duwaliy~+K)) (PUNC "))) (PUNC .))
S	سواء عن طريق المساعدات أو من خلال تونس نجحت الاستثمارات.	سافر الرجل من تونس إلى مصر.
T	najaHat tuwnis+u sawA'+N Ean Tariyq+i Al+musAEad+At+i CONJ >awo min xilAl+i Al+{sotivomAr+At+i .	sAfar+a Al+rajul+u min tuwnis+a <ilaY miSr+a .
G	Tunisia succeeded, either through aids or through investments.	The man traveled from Tunisia to Egypt.
P	(S (VP (PV najaHat) (NP (NOUN tuwnis+u) (PP (PP (NP (NOUN sawA'+N)) (PREP Ean) (NP (NOUN Tariyq+i) (NP (NOUN Al+musAEad+At+i)))) (CONJ >awo) (PP (PREP min) (NP (NOUN xilAl+i) (NP (NOUN Al+{sotivomAr+At+i)))) (PUNC .))	(S (VP (PV sAfar+a) (NP (NOUN Al+rajul+u)) (PP (PP (PREP min) (NP (NOUN tuwnis+a))) (PP (PREP <ilaY) (NP (NOUN miSr+a))))) (PUNC .))
S	الخطر ليس فقط في أن الاقتصاد ينهار.	تحدث كما كان دائما.
T	Al+xaTar+u layosa faqaT fiy >an~a Al+{iqotiSAd+a ya+nohAr+u .	-taHad~av+a -ka-mA kAn+a dA}im+AF .
G	The danger is not only in the collapsing economy.	He talked as he was always.

P	(S (NP (NOUN Al+xaTar+u)) (VP (PV layosa) (NP (-NONE- *T*)) (PP (ADVP (ADV faqaT)) (PREP fiy) (SBAR (CONJ >an~a) (S (NP (NOUN Al+{iqotiSAd+a)) (VP (IV ya+nohAr+u) (NP (-NONE- *T*))))))) (PUNC .))	(S (VP (PV -taHad~av+a) (NP (-NONE- *)) (PP (PREP -ka-) (SBAR (WHNP (PRON -mA)) (S (VP (PV kAn+a) (NP (-NONE- *T*)) (NP (ADJ dA}im+AF))))))) (PUNC .))

Table 1. The examples of the Arabic sentences parsed according to the annotation of the ATB

In Table 1 above, the meanings of the symbols S, T, G and P are respectively as follows: the Arabic Sentences, the Buckwalter Transliterations[1] of the sentences, their Gloss in English language and their Parsing representations according to the ATB annotations [11]. In the following, we present some structure examples extracted from these sentences to explain the application of different syntactic properties of the GP formalism. In particular, we focus, as simple choice, on the PP (Prepositional Phrase) structures to describe each property type.

3.1 The linearity property

The relation PREP ≺ NP states that a preposition (PREP) should always precede the Nominal Phrase (NP) in Arabic. Several following examples of the PP structures from Table 1 prove this:

	PREP constituents	**NP constituents**
1	Fiy	(**NP** (NOUN tuwnis+a)) (**ADJP** (ADJ waHod+a-) (NP (PRON -hA)))
2	Fiy	(**NOUN** xArij+i-) (**NP** (PRON -hA))
3	Ean	(**NOUN** Tariyq+i) (**NP** (NOUN Al+musAEad+At+i))
4	Min	(**NOUN** xilAl+i) (**NP** (NOUN Al+{sotivomAr+At+i))
5	Min	(**NOUN** tuwnis+a)
6	<ilaY	(**NOUN** miSr+a)

For a clearer tree representation, we showed in bold the constituents of each syntactic category, specified in a certain property describing PP. However, we can find from the same table a counter-example, that confutes this relation and shows the NP preceding the PREP, such as the part of PP structure " سواء عن " (sawA'+N Ean / either about):

NP constituents	**PREP constituents**
(**NOUN** sawA'+N)	Ean

Due to one counter-example, such relation should not be shown in the GP to build.

[1] Arabic Transliteration Table on Tim Buckwalter site: www.qamus.org/transliteration.htm

3.2 The adjacency property

Another relation type between the same categories PREP and NP can be noted in the examples of the PP structures presented for the linearity property. This is the adjacency property. Indeed, the PREP is always just after or just before the NP. These two categories can be adjacent if no counter-example is found. However, there is this example of the PP structure below. It shows the Left-Hand Side (LHS) of a rule example, which is, of course, PP and its Right-Hand Side (RHS), which is the structure "إلى" تحالف دولي" (<ilaY " taHAluf+K duwaliy~+K " / to an "international coalition"):

LHS	**RHS**			
PP	**PREP**	**PUNC**	**NP**	**PUNC**
1	<ilaY	"	(**NOUN** taHAluf+K) (**ADJ** duwaliy~+K)	"

This PP structure separates these two categories (PREP and NP) by a punctuation symbol ("), so the adjacency relation between PREP and NP cannot be valid.

3.3 The uniqueness property

This is a unary relation, so it concerns only one category. It induces each unique constituent in the described syntactic category. In the PP, we have many unique categories such as PREP, SBAR (Subordinate clause), ADVP (Adverbial phrase) and ADJP (Adjectival Phrase). Some examples of PP structures proving this uniqueness are shown below for some unique constituents in PP:

Unique constituents	**PREP**	**SBAR**	**ADVP**
LHS	**RHS**		
PP	(**ADVP** (ADV faqaT)) (**PREP** fiy) (**SBAR** (CONJ >an~a) (S (NP (NOUN Al+{iqotiSAd+a)) (VP (IV ya+nohAr+u) (NP (-NONE- *T*)))))		
	(**PREP** -ka-) (**SBAR** (WHNP (PRON -mA)) (S (VP (PV kAn+a) (NP (-NONE- *T*)) (NP (ADJ dA}im+AF)))))		
	(**PREP** fiy) (**NP** (NOUN xArij+i-) (NP (PRON -hA)))		

The first example of the PP structure is common for all specified constituents (PREP, SBAR, ADVP) because it shows them all unique. The second example is, by contrast, common for only PREP and SBAR. The NP could have also been unique in the PP structures if we do not have the PP structure example "اتدسواء عن طريق المساع" (sawA'+N Ean Tariyq+i Al+musAEad+At+i / either through aids) shown in the following:

LHS	RHS
PP	(**NP** (NOUN sawA'+N)) (**PREP** Ean) (**NP** (NOUN Tariyq+i) (NP (NOUN Al+musAEad+At+i)))

3.4 The obligation property

This relation is also unary. It concerns the constituents always presented, and which form a head in the described syntactic category. From the most of the PP structure examples, we can note that the PREP is a mandotary constituent in the PP. This relation can not be valid for the PP structure example " من تونس إلى مصر" (min tuwnis+a <ilaY miSr+a / from Tunisia to Egypt) shown in the following:

LHS	RHS
PP	(**PP** (PREP min) (NP (NOUN tuwnis+a))) (**PP** (PREP <ilaY) (NP (NOUN miSr+a)))

In this example, the PREP is not directly a mandatory constituent in the derivation subtree of the PP on the top, because this PP structure is composed of two successive PP that each one contains a PREP.

3.5 The requirement property

A requirement property indicates that the appearance of the first constituent involves the appearance of the second one. The relation ADVP ⇒ PREP respects this condition in many PP structure examples, such as the structure "أقتصاد الرهن فقط في ينفق" (faqaT fiy >an~a Al+{iqotiSAd+a ya+nohAr+u / only in the collapsing economy):

LHS	RHS
PP	(**ADVP** (ADV faqaT)) (**PREP** fiy) (**SBAR** (CONJ >an~a) (S (NP (NOUN Al+{iqotiSAd+a)) (VP (IV ya+nohAr+u) (NP (-NONE- *T*)))))

However, there are another PP structure example confuting this relation, which is "ليس أ لا في تونس وحدها بل في خارجها أيضا" (layosa fiy tuwnis+a waHod+a-hA, balo fiy xArij+i-hA >ayoD+AF / not only in Tunisia but also abroad), shown in the following:

LHS	RHS
PP	(**PP** (PRT (PART layosa)) (PREP fiy) (NP (NP (NOUN tuwnis+a)) (ADJP (ADJ waHod+a-) (NP (PRON -hA))))) (**PUNC** ,) (**CONJ** balo) (**PP** (PREP fiy) (NP (NOUN xArij+i-) (NP (PRON -hA)))) (**ADVP** (ADV >ayoD+AF))

In this example, the ADVP appears without the PREP. Furthermore, if we do not have this counter-example, this relation becomes valid. However, its symmetric relation

(PREP ⇒ ADVP) is not necessarily valid. The most famous structure of the PP (PREP NP), in effect, confutes it. This PP structure does not allow NP ⇒ PREP as valid relation. This is because there are PP structures where the PP appears instead of the PREP with the NP.
The relation SBAR ⇒ PREP is, however, always valid, so, in any PP structure example, if we have an SBAR, we find absolutely a PREP.

3.6 The exclusion property

The exclusion property is the opposite of the requirement one. It prevents the co-occurrence of two constituents. For the PP, when we observe the categories ADVP and SBAR in many Arabic examples of the PP structures, we can note that they do not appear together in these structures. Here are two examples proving this observation, which are "ليس أ لـ خارجها بل في تونس وحدها في فايضا" (layosa fiy tuwnis+a wa-Hod+a-hA, balo fiy xArij+i-hA >ayoD+AF / not only in Tunisia but also abroad) and "دائ كان دائما" (-ka-mA kAn+a dA}im+AF / as we was always):

	LHS	RHS
Only **ADVP**	**PP**	(**PP** (PRT (PART layosa)) (PREP fiy) (NP (NP (NOUN tuwnis+a)) (ADJP (ADJ waHod+a-) (NP (PRON -hA))))) (**PUNC** ,) (**CONJ** balo) (**PP** (PREP fiy) (NP (NOUN xArij+i-) (NP (PRON -hA)))) (**ADVP** (ADV >ayoD+AF))
Only **SBAR**		(**PREP** -ka-) (**SBAR** (WHNP (PRON -mA)) (S (VP (PV kAn+a) (NP (-NONE- *T*)) (NP (ADJ dA}im+AF)))))

If we are restricted to these examples, you will see that we have an exclusion relation between the ADVP and the SBAR (ADVP ⊗ SBAR). Unlike the requirement properties, this exclusion property is symmetric. Thus, when the SBAR appears, the ADVP do not. However, by expanding our vision on the PP structure example " فقط في أن ارهالاقتصاد ين" (faqaT fiy >an~a Al+{iqotiSAd+a ya+nohAr+u / only in the collapsing economy) where both SBAR and ADVP appears in the PP structure, this relation becomes invalid.

LHS	RHS
PP	(**ADVP** (ADV faqaT)) (**PREP** fiy) (**SBAR** (CONJ >an~a) (S (NP (NOUN Al+{iqotiSAd+a)) (VP (IV ya+nohAr+u) (NP (-NONE- *T*)))))

This linguistic study gives us an idea of the difficulty facing us to determine and enumerate the different valid syntactic relations in the Arabic language. These relations are implicit in the Arabic texts. Making them explicit, thanks to the GP formalism, can be useful for many NLP applications and different domains, such as the ambiguity resolution, the probabilistic parsing and the dependency grammar building.

Formalization phase

As we have already mentioned in Section 1, the elaboration of a solid and detailed enrichment method cannot be directly made without modeling the tools to use as input. In our case, this means that we have to generate specific formalizations to the treebank ATB and the GP. This facilitates and clarifies better the enrichment method.

First, we present the description of the CFG (Context-Free Grammar), which is composed of a set of production rules (constructions). The latter are used to produce structures of words. Formally, it is defined by the 4-tuple G= (N, Σ, P, S) where: N is a finite set of non-terminal symbols, Σ is a finite set of terminal symbols, P is a finite set of rules formed as $\alpha \rightarrow \beta$ with $\alpha \in N$ and $\beta \in (N \cup \Sigma)^*$ and $S \in N$ is the start symbol. The formal language of G is then defined as $L(G)=\{w \in \Sigma^* \mid S \vdash^* w\}$. For each derivation of S, w corresponds to a tree t_w. In natural languages, w corresponds to a sentence Sent, which is associated to a tree t_{Sent} according to the grammar G.

On the one hand, the ATB, as a corpus of manually annotated sentences of a natural language (Arabic) can be seen as a sequence of pairs (Sent, t_{Sent}). So, it is defined by $TB=\{(Sent, t_{Sent}) \mid S \vdash^* Sent\}$ where Sent is a sequence of Arabic words, giving a complete meaning. So, $Sent \in M^*$ where M is the set of the treebank words. However, $t_{Sent} \in \mathscr{A}$ where $\mathscr{A}$ is the set of trees given by parsing each treebank sentence Sent according to G. As the annotations given by the ATB are extended to several analysis levels (word, phrase and sentence levels), this improves the definition of the ATB to be a 7-tuple TB = (M, Ψ, P, $\mathscr{T}$, Ω, $\mathscr{S}$, $\mathscr{A}$). M is a set of treebank words. $\Psi = \psi_1 \times \psi_2 \times ... \times \psi_n$ is an n-tuple of sets ψ_i of information types (morphological, syntactic and semantic). The latter specify the corpus words with the form $(c_1, c_2, .., c_n) \in \Psi$ where c_i is an information of a defined type i of n information types (e.g. lexical category, transliteration, gloss). P is the treebank phrase set (a phrase is a sequence of words giving elementary meaning) as $p \in M^*$. $\mathscr{T}$ is the elementary tree set t_p given from parsing phrases $p \in P$. $\Omega = \omega_1 \times \omega_2 \times \ldots \omega_z$ is an n-tuple of sets ω_j of information types. The latter specify the corpus phrases with the form $(d_1, d_2, .., d_z) \in \Omega$ where d_j is an information of a defined type j of z information types (e.g. syntactical category, hashtag). $\mathscr{S}$ is the set of sentences as $Sent \in M^*$. $\mathscr{A}$ is the complete tree set obtained from parsing sentences $Sent \in \mathscr{S}$.

On the other hand, the GP is a grammar that defines a set of relations between grammatical categories not in terms of production rules (like CFG) but in terms of local constraints (so-called properties). As we specified in the previous section, the syntactic properties describe linguistic phenomena between constituents such as linear precedence (≺), mandatory co-occurrence (⇒), restricted co-occurrence (⊗), obligation (oblig), uniqueness (unic) and adjacency (±). Formally, this grammar can be defined by a 3-tuple G'= (N, Σ, R). N is a finite set of syntactic categories. Σ is a finite set of lexical categories. R is a finite set of syntactic properties that links $\forall \alpha \in N$ to $\forall \beta_1$ and $\beta_2 \in (N \cup \Sigma)$ in any of the following 6 ways: $\alpha: \beta_1 \prec \beta_2$, $\alpha: \beta_1 \pm \beta_2$, $\beta_1 \in unic(\alpha)$, $\beta_1 \in oblig(\alpha)$, $\alpha: \beta_1 \Rightarrow \beta_2$, $\alpha: \beta_1 \otimes \beta_2$. We deduce each of these properties from the set P defined in G.

Now, as the needed tools to use are formally modeled, it is necessary to know how to integrate them to succeed the enrichment method. We may consider this enrichment for the ATB phrases as a satisfaction verification of properties provided from the GP. It can be a Constraint Satisfaction Problem (CSP). Formally, we can model this problem by the 5-uplet TBG = (S(TB), S(G'), Const(TB), Const(G'), Prop(G')) where:

- S(TB) = $\{p_1, p_2, \ldots p_n\}$ = P is a finite set of the ATB phrases.
- S(G') = $\{t_1, t_2, \ldots t_m\}$ = N is a finite set of the GP syntactic categories.
- Const(TB) = $i=1nConst(pi)$ where Const(p_i) = $\{c_{i1}, c_{i2}, \ldots c_{ie}\}$: set of the words of p_i, label(c_{ix}) is its grammatical category (label(c_{ix}) is equal to $c_1(c_{ix})$ for lexical category or to $d_1(c_{ix})$ for syntactic category).
- Const(G') = $j=1mConst(tj)$ where Const(t_j) = $\{c_{j1}, c_{j2}, \ldots c_{jf}\}$: set of the constituents (grammatical categories) of the syntactic category t_j in the GP.
- Prop(G') = $j=1mProp(tj)$where Prop(t_j)=[Prop_const(t_j), Prop_unic(t_j), Prop_oblig(t_j), Prop_lin(t_j), Prop_adjc(t_j), Prop_exig(t_j), Prop_excl(t_j)], for example, Prop_lin(t_j) = $\{p_{j1}, p_{j2}, \ldots p_{jg}\}$ is the linearity property set describing t_j in the GP. Each p_{jk} (with $1<k<g$) is a relation between two elements c_{jx} and $c_{jy} \in$ Const(t_j) (p_{j1}=tj:$c_{jx} \prec c_{jy}$).

In order to solve this issue, we need, in the first instance, to look in the GP for the syntactic category of each ATB phrase. Formally, for each phrase $p_i \in$ P in the ATB, we search in the GP for its syntactic category $t_j \in$ N where label(p_i) = t_j. The set of properties Prop(t_j) describing t_j will be used to enrich p_i by verifying the satisfaction of these properties. As a result, this problem would formally be solved.

The enrichment method of the ATB

Now that the formalization phase is totally accomplished, it became possible to represent, in detail, the other phases of the enrichment method, which would be written in algorithms. Note that these phases are based on the enrichment idea of the French treebank FTB, where the properties were proposed by [6]. For clarity, Figure 1 shows our ATB enrichment method, which consists of three main phases: formalizing the problem, inducing the GP from the ATB and regenerating the latter with a new syntactic property-based representation.

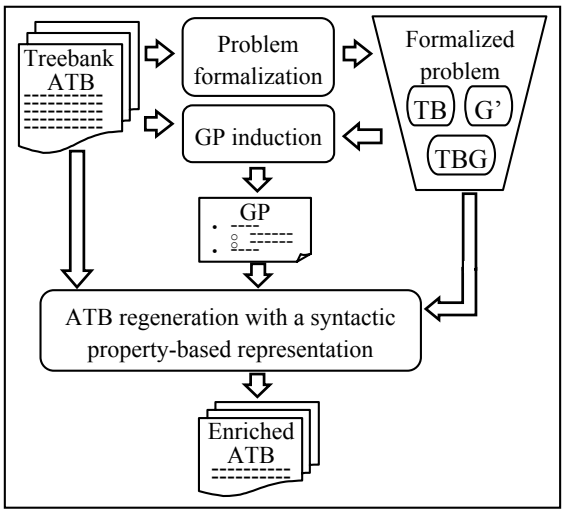

Fig. 1. The ATB enrichment method

We chose to devote an entire section (the previous one) to explain the first phase, the formalization, as its important role in our enrichment method and particularly in this paper.

The second phase, the GP induction, is already applied by [4], which produced an Arabic GP. In this phase, the GP is constructed automatically from the ATB. This directive is more favorable than building the GP manually. The latter is more challenging and expensive. It needs to use a corpus, which contains all the rules of the Arabic grammar. This increases the GP development time and requires the collaboration of several linguists. Therefore, the obtained GP was automatically induced and independently of the language and source treebank formalism. That is why the GP is not directly generated from the ATB, but rather from a CFG (as shown in Figure 2).

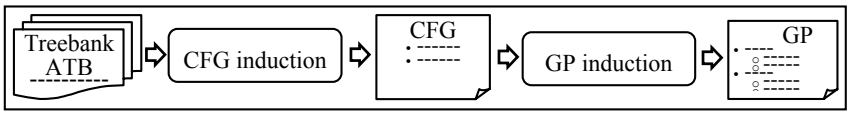

Fig. 2. The induction phase of the GP from the ATB

For more details, the CFG induction step involves the generation of the set of all the possible assignments for each syntactic category represented in the ATB. This set will be used in the GP induction step to generate the set of properties associated to this syntactic category. All of the GP properties are described except the dependency ones. Adjacency properties are added to this set. They concern the direct order relation between two constituents of the syntactic category. The induction mechanism provides also a control of the granularity level of the categories in order to compromise between quantity and quality of these categories. This control represents each category on feature structures related to hierarchy types. The obtained GP is robust not only because of the power of the GP formalism but also thanks to the qualities

inherited from the ATB. For instance, it has rich annotations and a consistent representation structure to the GP one [5].

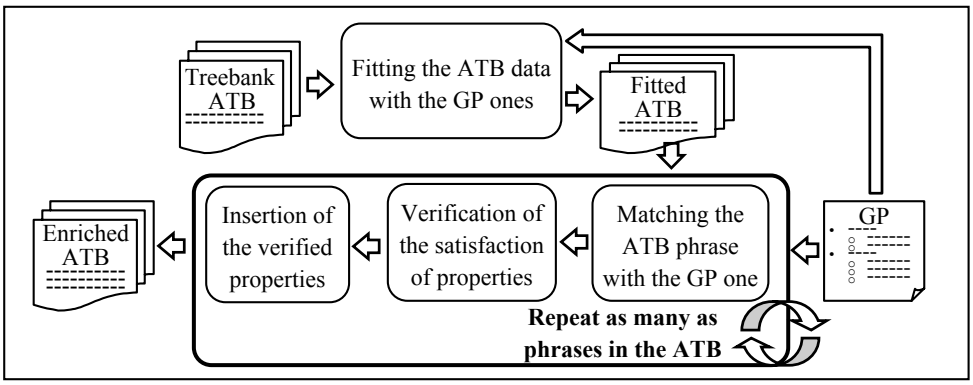

Fig. 3. The ATB regeneration phase with a syntactic property-based representation

The obtained GP is used as an input for the ATB regeneration phase with syntactic properties. As mentioned in Figure 3, this phase is based on many steps: the first is a fitting step of the ATB data according to the GP one. The three other steps relate to each phrase in the ATB. Therefore, we need, for all the ATB phrases, to follow a browsing mechanism through the ATB sentences. This is to check for each phrase of each sentence, the satisfaction of the properties describing its syntactic category in the GP. To check this satisfaction, it needs to use previously developed constraint solvers. In the following sub-sections, we explain the different steps of this phase.

5.1 Fitting the ATB Data with the GP ones

Since our goal is to enrich the ATB with syntactic properties, it should have a data structure able to host this new information. The parenthesis format "penntree" of the ATB does not provide this structure, and requires preparing its data in a suitable format. The format "xml" can form this structure. To achieve this, we have made a conversion recursive process of encountered open and close parentheses in the format “penntree” to xml tags. In addition, if the ATB category granularity was modified, we would include a verification model of matches in this step to replace the ATB raw categories with categories whose granularity is modified.

The following steps are encapsulated in a browsing mechanism repeated as many as phrases in the ATB. As a result, we will have the fitted ATB as an input, able to host GP syntactic properties. The output is a new version of the ATB, which is enriched with these verified properties as satisfied or not.

5.2 Matching an ATB phrase with a GP one

The matching between the ATB and the GP consists of browsing the ATB, phrase by phrase, and for each one, searching for the correspondent in the GP of its category.

The properties describing the found correspondent will be verified and will enrich the current ATB phrase. Formally, in order to match between an ATB phrase $p_i \in P$ and the correspondent of its category in the GP, we need, for each $\alpha \in N$ in the GP, to look for the case where the p_i category label(p_i) is equal to α (label(p_i) = α).

5.3 Verification of the satisfaction of the properties

This step is the heart of the enrichment method. It verifies the satisfaction of the properties, which describes a GP category matched with the ATB phrase. Formally, we just need to verify, for each ATB phrase p_i (with t_j =label(p_i)), the satisfaction of all the properties of Prop(t_j) obtained from the GP. We have used for that a set of methods to check the satisfaction of the properties. Each method, so-called "constraint solver", verifies if a given Arabic phrase tagged with a specific syntactic category respects a given property, which describes this syntactic category in the GP. The solution produced by a solver is the result of this verification (property satisfied or violated). Then, we associate this solution to the p_i description. As these constraint solvers play an importance role in our method, we have chosen to devote an entire section (the next one) to introduce their descriptions.

5.4 Insertion of the verified properties

This task adds to each ATB phrase the result of the verification (either satisfied or not) of the properties that describe its category. The insertion is done by using a new tag that combines the ATB and the GP. However, this enrichment makes too large the new ATB size. It is due to the exponential increase of the number of the new tags with the number of properties in each phrase of the GP.

Descriptions of the constraint solvers

Let us assume that the matching has been achieved, so the current ATB phrase category is equal to the found GP one. The descriptions of these solvers, introduced in the following, are inspired from the interpretations of [6]. Let us first define some variable definitions to use in these descriptions.

p: the given Arabic phrase.
Const(p): constituent set of the ATB phrase **p.**
Const(t): set of the constituents of the GP category **t** (where **t=label(p)**).
fd: boolean, returns true if **c** is found.
label(c): grammatical category of the word or the phrase **c.**
nb_intersect: number of constituents in the intersection between **Const(p)** and **Const(t)**
verif: string ("+" or "-").
nb_occ: number of occurrences of a constituent in **Const(p).**
type_p: property by type between two constituents $\mathbf{c_x}$ and $\mathbf{c_y}$

of **Const(t)**, **type_p** contains only c_x for unary property type (uniqueness, obligation).
v_type_p: verified property by type (constituency (const), linearity (lin), adjacency (adjc), uniqueness (unic), obligation (oblig), requirement (exig), exclusion(excl)) (has "+" if satisfied, "-" if not). Firstly, **v_type_p** is empty (← NIL) and may remain if the constituents of **type_p** is not found in **p.**
verifProp(): method to create a verified property.

6.1 The solver of constituency properties

This solver verifies the consistency between the categories of the constituents of the current ATB phrase and the constituents of its GP correspondent. This is to ensure that the intersection of these two sets really includes all the words of the ATB phrase.

```
Input: Const(p), Const(t), nb_intersect ← 0,
const_p
Output: v_const_p
for each ca in Const(p), do
   for each cb in Const(t), do
      if label(ca)= cb, then
         nb_intersect ← nb_intersect + 1
if nb_intersect = card(Const(p), then
   v_const_p ← verifProperty(p, Const(p), "+")
else
   v_const_p ← verifProperty(p, Const(p), "+")
return v_const_p
```

This algorithm browses Const(p) and verifies that the category of each element is included in Const(t). The value of nb_intersect is then incremented. If it is equal to the Const(p) cardinal, the property is considered then as satisfied.

6.2 The solver of the linearity properties

This solver checks the current linearity property of the GP syntactic category. This property is satisfied only if the two constituents of this category in that relation are also found in the given phrase and that the first constituent precedes the second one.

```
Input: Const(p), lin_p, v_lin_p ← NIL
Output: v_lin_p
for each ca in Const(p), do
  if label(ca)= lin_p.cx, then
    for each label(cb) in Const(p), do
     if a≠b and label(cb)= lin_p.cy, then
```

```
      if a>b, then
       v_lin_p ← VerifProperty(p, lin_p, "-
")
      else
       v_lin_p  ←  VerifProperty(p,  lin_p,
"+")
return v_lin_p
```

This algorithm browses the categories of the Const(p) set elements to search for the two distinct linear constituents c_x and c_y of the GP category t and verifies that the position of the first is not greater than the position of the second one.

6.3 The solver of the adjacency properties

This solver checks the current adjacency property of the syntactic category in the GP. The satisfaction is ensured only if the two adjacent constituents of this category exist in the given phrase and the first is directly before or after the second one.

```
Input: Const(p), adjc_p, v_adjc_p ← NIL
Output: v_adjc_p
for each c_a in Const(p), do
 if label(c_a)= adjc_p.c_x, then
  for each c_b in Const(p), do
   if a≠b and label(c_b)= adjc_p.c_y, then
      if a≠b-1 and a≠b+1, then
       v_adjc_p ← VerifProperty(p, adjc_p, "-")
      else
       v_adjc_p ← VerifProperty(p, adjc_p, "+")
return v_adjc_p
```

This algorithm browses the set Const(p) to look for the two adjacent constituents c_x and c_y of the GP category t and verifies that the second is neither indirectly before nor after the first one. We use the symbol "±" in the adjacency relation.

6.4 The solver of the uniqueness properties

This solver checks the current uniqueness property of the current syntactic category in the GP. The satisfaction is reached if the constituent of this property (in case it has been found) appears only once in the given phrase.

```
Input: Const(p),unic_p,nb_occ←0,v_unic_p ← NIL
Output: v_unic_p
for each c_a in Const(p), do
 if label(c_a)= unic_p.c_x, then
    nb_occ ← nb_occ +1
```

```
if nb_occ ≥1, then
 if nb_occ = 1, then
    v_unic_p ← verifProperty(p, unic_p, "+")
 else
    v_unic_p ← verifProperty(p, unic_p, "-")
return v_unic_p
```

This algorithm browses the set Const(p) to search for the constituent unic_p of t and verifies that its cardinality nb_occ is not greater than 1.

6.5 The solver of the obligation properties

This solver checks the current obligation property of the found GP category. This property is satisfied if the constituent "head" of this category is found in the treebank phrase.

```
Input:  Const(p),oblig_p,fd  ←  false,v_oblig_p  ←
NIL
Output: v_oblig_p
for each ca in Const(p), do
 if label(ca)= oblig_p.cx, then
    fd ← true
    break
if fd = true, then
 v_oblig_p ← verifProperty(p, oblig_p, "+")
else
 v_oblig_p ← verifProperty (p, oblig_p, "-")
return v_oblig_p
```

This algorithm browses Const(p) to search for the obligatory constituent oblig_p of the GP category t. If the algorithm find it, the variable "found" will return true.

6.6 The solver of the requirement properties

This solver checks the current requirement property of the current syntactic category in the GP. The satisfaction is ensured only if, when the constituent involving another in this property, is found in the given phrase, the involved one is also found.

```
Input: Const(s), fd ← false, exig_p, v_exig_p ←
NIL
Output: v_exig_p
for each ca in Const(s), do
 if label(ca)= exig_p.cx, then
   fd ← false
   for each cb in Const(s), do
```

```
    if a≠b and label(c_b)= exig_p.c_y, then
     v_exig_p ← verifProperty(s, exig_p, "+")
     fd ← true
     break
   if fd =false then
    v_exig_p ← verifProperty(s, exig_p, "-")
    break
return v_exig_p
```

This algorithm browses the set Const(p) to search for the two constituents c_x and c_y of the GP category t in a requirement relation and verify that, if the first constituent is found in Const(p), then the second one should exist in Const(p).

6.7 The solver of the exclusion properties

This solver checks the current exclusion property of the syntactic category in the GP. This property is satisfied only if its constituents do not appear both in the given phrase.

```
Input:  Const(s),excl_p,verif  ←  "+"  ,v_excl_p  ←
NIL
Output: v_excl_p
a ← search(excl_p.c_x, Const(s))
if a > 0, then
 for each c_b in Const(s), do
  if a≠b and label(c_b)= excl_p.c_y, then
   verif ← "-"
   break
v_excl_p ← verifProperty(s, excl_p, verif)
return v_excl_p
```

This algorithm browses the set Const(p) to look for the constituents c_x and c_y of t in an exclusion relation and mark it as satisfied if they are both not found or only one of them is found in p. So, in all cases there would be a verified property in return. We have used the method search() to search only for the position of c_x in the categories of Const(p).

Experimentation and evaluation

We have tested our method on the ATB corpus (ATB2v1.3 version), which includes 501 stories from the Ummah Arabic News Text. The latter contains 144,199 words before the clitic-separation. As we have already mentioned in the previous section, we need to have the ATB in a "xml" format, as input of the property verification task. Having such format, we had not exempted from preparing a simple version in the fitting step due to the handling difficulty of the available version. We have used more

specifically the "penntree" format (the vowelized version) to convert it into "xml". We have induced the GP from only the half of the ATB in order to make the study corpus different to the test one. In what follows, we will present some of the obtained results after citing above the meanings used in the headers of the tables due to lack of space:

#	Frequency	#C	Number of possible constituents
∑	Total	#R	Number of production rules
XP	Phrase		
#P	Number of properties		

Table 2. The header meanings

First, we have found that the ATB is composed of 841 grammatical categories of which 348 are syntactic (put in 21 phrase groups). Table 3 shows the distribution of the ATB phrase by frequency, the possible constituent number and the production rule number.

XP	#	# C	# R	XP	#	# C	# R	XP	#	# C	# R
NP	110748	299	4824	PRT	2292	13	14	PRN	65	10	20
PP	22100	22	263	ADVP	539	6	68	LST	56	2	2
S	19358	138	1230	NAC	221	18	53	SQ	51	12	26
VP	15947	342	6675	FRAG	178	22	56	CONJP	37	3	2
SBAR	9524	47	380	WHADVP	136	3	34	INTJ	11	1	1
WHNP	4574	3	64	UCP	132	19	88	X	5	4	5
ADJP	3665	88	593	SBARQ	68	19	51	WHPP	3	3	3
								∑	841		14452

Table 3. The Distribution of the phrases in the ATB

From Table 3, we may notice that the most frequent phrase in the ATB is the Nominal Phrase (NP). It even contains large numbers of possible constituents and production rules (equals to 1/3 of all rules). This dominance does not excessively influence the distribution of the properties. According to Table 4 showing information about the 10 most frequent phrases (in the lowest granularity level), NP has the greatest numbers of uniqueness (40%) and exclusion (83%) properties. However, the leader in this distribution becomes the VP (Verbal Phrase) for the linearity properties (62%) and the SBAR (subordinate clause) for of the requirement ones (53%). We may also note that dealing with such high frequencies of uniqueness properties for most phrases implies the need to have a unique constituent in each Arabic phrase. The linear order is important to the VPs as to the SBARs. For the constituency properties, we have applied them once for each phrase. Their frequency is then equal to the phrase frequency.

Regardless to the given property distribution, we obtained an important and varied implicit information in such Arabic text. We may give some examples: In the Arabic VP, we have the linearity property IV≺PP, which requires that the PP (Propositional Phrase) must never precede the IV (Imperfect Verb). Similarly, we have the requirement property ADJ⇒NP, which needs the presence of a NP if an ADJ (adjective) exists.

XP	Uniqueness		Linearity		Requirement		Exclusion		∑	
	#P	#	#P	#	#P	#	#P	#	#P	#
NP	22	21686	50	1237	6	125	404	44742192	483	44875988
PP	12	1840	17	1795	15	1947	106	2342600	151	2370282
S	11	539	45	3807	12	89	99	1916442	168	1940235
VP	19	16694	104	26536	16	1493	196	3125612	336	3186282
SBAR	13	5238	30	9053	11	4913	129	1228596	184	1257324
WHNP	5	4574	2	4	2	4	8	36592	18	45751
ADJP	10	88	17	68	12	88	87	318855	127	322764
PRT	12	2290	0	0	0	0	66	151272	79	155854
ADVP	6	559	3	21	4	22	12	6468	26	7609
NAC	9	426	9	189	10	204	33	7293	62	8333
∑	162	54721	346	43096	139	9279	1288	53891401	1958	53998567

Table 4. The distribution of the ATB properties by phrase

By focusing on the distribution of the property types, it can be seen that the parts of the obligation and the adjacency properties are virtually zero. The obligation ones have only 3 properties (describing the following 3 phrases: LST, INTJ and WHPP) with 70 occurrences. The adjacency ones do not have any properties. This shows that we do not need to have neither mandatory constituent (head) in the most of the phrases nor any condition about a direct order between constituents of the same phrase. This proves the variety of structuration of the phrase rules in Arabic.

For an overview of all the property types, we can observe their high frequency compared to other enriched treebanks (e.g. the FTB) [6]. Indeed, according to Table 5, only the obligation properties are negligible in the ATB. The others vary from tens of thousands to millions. This large number can go greater if we extend our work to the highest granularity level of grammatical categories in the ATB. This reflects the richness and the variety of the structures in Arabic.

Treebanks	Uniqueness	Obligation	Linearity	Requirement	Exclusion	∑
ATB	54721	70	43096	9279	53891401	53998567
FTB	38007	32602	27367	11022	89293	198291

Table 5. The distribution of the properties in treebanks

In such phrases, it is also possible to know the most frequent types of properties. The following Figure 4 represents the distribution of the ATB properties by type (only those with comparable values). Thus, it is important to know that many categories can be absent but not repeated in Arabic phrases. This finding is based on the big difference between uniqueness frequencies of properties and obligation ones. It is also clear that we have a great abundance of the exclusion properties versus a virtual absence of the obligation ones. This wide gap needs to be adjusted by describing new interpretations to these types. The adjacency properties however cannot have another conception because it concerns an order in which information is automatically defined.

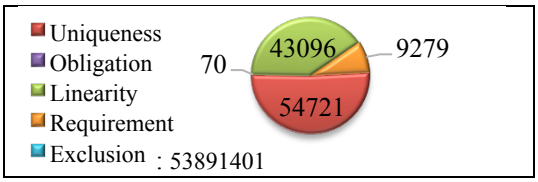

Fig. 4. The distribution of the ATB properties by type

As already mentioned in the previous section, the verification of the property satisfaction distinguishes those satisfied from those violated. We represent in Table 6 the satisfaction rates of the properties by type of the phrases VP as instance.

Property state	Uniqueness		Linearity		Requirement		Exclusion	
	#	%	#	%	#	%	#	%
Satisfied	15932	99.91	26529	99.99	1491	99.87	3125590	99.99
Violated	15	0.09	7	0.01	2	0.13	22	0.01

Table 6. The satisfaction rates of the VP property types

According to the obtained results, the number of violated properties is negligible compared to satisfied ones. If the ATB is considered as a large coverage resource, the used GP in the enrichment task inherits also this richness.

The property distribution can be detailed to finer levels by describing individually each property of such phrase and type. This description let us to know which property is more important. We may not have a precise information about the importance of the properties when we are restricted on defining the property distribution by type. Thus, this distribution in a same type can be no homogenous. We present in Figure 5, for example, the distribution of the VP properties to determine the most frequent ones. The abscise axes of the shown schemes are the property indexes and the ordinate ones are their frequencies. We may consider in that case that the most frequent properties have the highest weight (occurrence number). So, it can be admitted as relevant information. We fix that a strong property have at least 1500 occurrences for the uniqueness and the linearity properties and 500 occurrences for the requirement ones. Table 7 gives us the strong properties of the VP.

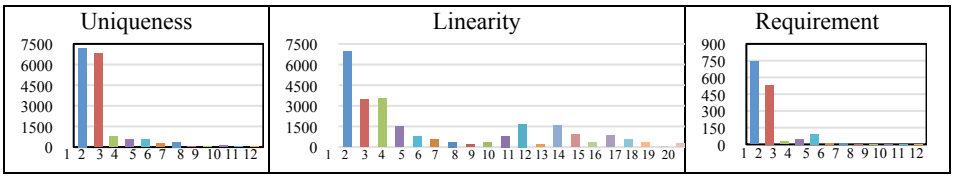

Fig. 5. The distribution of the VP properties

Using these results, we can automatically measure the property weights in such construction. This information can be included in the GP to ease the parsing process. Indeed, we can check the satisfaction only of the strong properties. The others can be relaxed. This information is also useful to evaluate the difficulty of the processing in

cognitive systems since the violation of a strong property implies the most important difficulty.

	Uniqueness	Linearity	Requirement
Index	1, 2	1, 2; 3; 11, 13	1, 2
Property	PV, IV	PV ≺{NP, PP}; IV ≺ PP; PRT ≺ {NP, IV}	NOUN ⇒NP, ADJ ⇒NP

Table 7. The strong properties of the VP

Conclusion and perspectives

We proposed in the present paper a method of treebank enrichment based on the GP formalism. According to the enrichment process, the efficiency of this formalism is proved both in terms of direct access on constraints as well as in terms of simple and local representation. We started with a linguistic study on Arabic syntactic structures. From these structures, we induced some syntactic properties describing the syntactic categories of these structures. This study verifies also the validity (satisfaction) of the deduced properties. The enrichment method is applied on a large corpus (the ATB) and gave us good results and various properties of different types.

As perspectives, in order to offer a very precise representation of the syntactic information in the ATB, we can enrich or improve the relation set presented in the induced GP. For example, proposing an interpretation of the dependency property or modifying the description of the obligation and exclusion properties. In future works, we can optimize our enrichment method by integrating several control mechanisms into the determination of the syntactic categories and into the verification of their properties. We can go further by applying our enrichment method to other annotated corpora obtained from existing parsers.

References

1. Abdul-Mageed, M., Diab, M.: AWATIF: A Multi-Genre Corpus for Modern Standard Arabic Subjectivity and Sentiment Analysis. Language Resources and Evaluation Conference (LREC'12), Istanbul, Turkey (2012)
2. Alkuhlani, S., Habash, N.: A Corpus for Modeling Morpho-Syntactic Agreement in Arabic: Gender, Number and Rationality. Association for Computational Linguistics (ACL'11), Portland, Oregon, USA (2011)
3. Alkuhlani, S., Habash, N., Roth, R.: Automatic Morphological Enrichment of a Morphologically Underspecified Treebank. North American Chapter of the Association for Computational Linguistics: Human Language Technologies (HLT-NAACL'13), pp. 460-470, Atlanta, Georgia, USA (2013)
4. Bensalem R. B., Elkarwi, M.: Induction d'une grammaire de propriétés à granularité variable à partir du treebank arabe ATB. Rencontre des Étudiants Chercheurs en Informatique pour le Traitement Automatique des Langues (RECITAL'14), pp. 124-135, ATALA, ACL-ontology, Marseille, France (2014)

5. Bensalem R. B., Elkarwi, M., Haddar, K., Blache, P.: Building an Arabic Linguistic Resource from a treebank: The Case of Property Grammar. Text, Speech and Dialogue (TSD'14), pp. 240-246, Springer, Czech Republic (2014)
6. Blache, P., Rauzy, S.: Hybridization and treebank enrichment with constraint-based representations. LREC'12- Workshop on Advanced Treebanking. Istanbul. Turkey (2012)
7. Çakıcı, R.: Automatic induction of a CCG grammar for Turkish. ACL Student Research Workshop, pp. 73–78, Ann Arbor, Michigan (2005)
8. El-taher, A. I., Abo Bakr, H. M., Zidan, I., Shaalan, K.: An Arabic CCG approach for determining constituent types from Arabic treebank. Journal of King Saud University - Computer and Information Sciences, pp. 1319-1578 (2014)
9. Habash, N., Rambow O.: Arabic Tokenization, Part-of-Speech Tagging and Morphological Disambiguation in One Fell Swoop. ACL, pp. 573-580, Ann Arbor, Michigan (2005)
10. Hovy, E., Marcus, M., Palmer, M., Ramshaw, L., Weischedel, R.: OntoNotes: The 90% Solution. North American Chapter of the Association for Computational Linguistics (NAACL'06), pp. 57–60, USA (2006)
11. Maamouri, M., Bies, A., Buckwalter, T., Mekki, W.: The Penn Arabic Treebank: Building a Large-Scale Annotated Arabic Corpus. NEMLAR Conference on Arabic Language Resources and Tools, Cairo, Egypt (2004)
12. Maruyama, H.: Structural Disambiguation with Constraint Propagation. In: ACL'90 Workshop on Dependency-based Grammars, pp.31-38. Pittsburgh, Pennsylvania, USA (1990)
13. Müller, H. H.: Annotation of Morphology and NP Structure in the Copenhagen Dependency Treebanks (CDT). International Workshop on Treebanks and Linguistic Theories, pp. 151-162, University of Tartu, Estonia (2010)
14. Oepen, S., Flickinger, D., Toutanova, K., Manning, C. D.: LinGO Redwoods - A Rich and Dynamic Treebank for HPSG. LREC'02 - workshop on parsing evaluation, Las Palmas, Spain (2002)
15. Palmer, M., Babko-Malaya, O., Bies, A., Diab, M., Maamouri, M., Mansouri, A., Zaghouani, W.: A Pilot Arabic Propbank. LREC'08, Marrakech, Morocco (2008)
16. Pollard, C., Sag, I.: Head-driven Phrase Structure Grammars. In: Chicago University Press (1994)
17. Tounsi, L., Attia, M., Van-Genabith, J.: Automatic Treebank-Based Acquisition of Arabic LFG Dependency Structures. The European Chapter of the ACL (EACL) Workshop on Computational Approaches to Semitic Languages, pp. 45–52, Greece (2009)