A Formal Modeling Method to Enrich the Arabic Treebank ATB with Syntactic Properties

Raja Bensalem Bahloul[1], Kais Haddar[1] and Philippe Blache[2]
[1]*Multimedia InfoRmation Systems and Advanced Computing Laboratory, Higher Institute of Computer Science and Multimedia, Sfax, Tunisia*
[2]*Laboratoire Parole et Langage, CNRS, Université de Provence, Aix-en-Provence, France*

Keywords: Formal Modelling, Treebank Enrichment, Arabic Language, Property Grammar.

Abstract: The enrichment of an Arabic treebank with syntactic properties can facilitate many types of parsing processes. This enrichment allows also the increase of its use in different NLP applications, the acquirement of new linguistic resources and the ease of the probabilistic parsing process by using statistics to limit the properties to the satisfied ones or to the most frequent ones. In this context, our proposed enrichment method is based on a formalization phase, a Property Grammar induction phase from a source treebank and a treebank regeneration phase with a new syntactic property-based representation. Starting with a formalization phase in our enrichment problem may succeed its resolution procedure. In fact, it limits the specification of the data sets and the interactions between them to the used ones, which avoids any duplication. The formalization allows also the anticipation of the constraints to respect in the problem. The implementation of this enrichment method is experimented essentially on the Arabic treebank ATB. This experiment provides us with good and encouraging results and various properties of different types.

1 INTRODUCTION

The formalization phase plays an important role to improve the performance of any problem. Thanks to this phase, only the data sets needed to use and their interactions will be specified, which implies the avoidance of duplication. The constraints to respect in the problem are also anticipated in this phase. In the case of a treebank enrichment issue, we may find many other motivations. Thus, treebanks are useful not only to gain other linguistic resources (e.g. CFGs, lexicons), but also to solve some grammatical ambiguities (Koller and Thater, 2010) and even to ease the probabilistic parsing process (Cahill, 2008). However, its large amount of data may make this process more complicated. In addition, treebanks do not give any information about syntactic structures such as form occurrences, essential parts and the order of constituents. This can reduce the exploitation of treebanks by many applications (e.g. parsers) or formalisms (e.g. GP, HPSG). The enrichment of treebanks with syntactic properties generated from a Property Grammar (GP) (Blache and Rauzy, 2012) may be a powerful alternative to remedy such deficiencies. By contrast, this enrichment is not an easy and direct task 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 into this context. Our goal is to propose a formalization phase that facilitates and clarifies the enrichment method of the Arabic treebank ATB with syntactic properties 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 a source Arabic treebank and the new treebank generation based on syntactic property.

Bahloul, R., Haddar, K. and Blache, P..
A Formal Modeling Method to Enrich the Arabic Treebank ATB with Syntactic Properties.
In *Proceedings of the 7th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2015) - Volume 2: KEOD*, pages 108-117
ISBN: 978-989-758-158-8

This paper is organized as follows: Section 2 is devoted to present some related works. Section 3 describes the formalization phase. Section 4 explains our enrichment method. Section 5 presents experimental results and discussions. Section 6 gives a conclusion and some perspectives.

2 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 (Blache and Rauzy, 2012) 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: linear order ($\prec$), obligation of co-occurrence ($\Rightarrow$), interdiction of co-occurrence ($\otimes$), dependency ($\rightsquigarrow$), interdiction of repetition (Unic) and head (Oblig).

The ATB (Maamouri et al., 2004) 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 (Müller, 2010) 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 et al., (Oepen et al., 2002) 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 (Çakıcı, 2005), who created CCGbanks by converting the syntax graphs in the Turkish treebank into CCG derivation trees.

For French, Blache and Rauzy (Blache and Rauzy, 2012) 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 (Hovy et al., 2006) and the Proposition Bank project (Propbank) for Arabic (Palmer at al., 2008) are some instances of treebank extensions. The latter incorporate semantic level annotations. The contribution of Alkuhlani and Habash (Alkuhlani and Habash, 2011) provides an enrichment, which adds annotations that models attributes of the functional gender, number and rationality. The work of Abdul-Mageed and Diab (Abdul-Mageed and Diab, 2012) has even touched the sentimental level by associating specific annotations to the ATB sentences. There is also the work of Alkuhlani et al., (Alkuhlani et al., 2013), but it enriches the Columbia Arabic Treebank (CATiB) with the most complicated POS tags and lemmas applied in the ATB (Maamouri et al., 2004).

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 (Habash and Rambow, 2005) with a TAG grammar, the Tounsi and al. contribution (Tounsi et al., 2009) with an LFG grammar and the El-taher et al., contribution (El-taher et al., 2014) 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. The GP is an approach extremely based on constraint satisfaction. By applying it to enrich the ATB, 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 (Bensalem et al., 2014). It is not manually built but automatically generated from an ATB part (Bensalem and Elkarwi, 2014).

3 FORMALIZATION PHASE

As we have already mentioned, 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.

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 ∈ M* where M is the set of the treebank words. However, $t_{Sent} \in \mathcal{A}$ where $\mathcal{A}$ 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, $\mathcal{T}$, Ω, $\mathcal{S}$, $\mathcal{A}$). 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 ∈ M*. $\mathcal{T}$ is the elementary tree set t_p given from parsing phrases p ∈ 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). $\mathcal{S}$ is the set of sentences as Sent ∈ M*. $\mathcal{A}$ is the complete tree set obtained from parsing sentences Sent ∈ $\mathcal{S}$.

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.
- $Const(TB) = \bigcup_{i=1}^{n} Const(p_i)$ 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).
- $Const(G') = \bigcup_{j=1}^{m} Const(t_j)$ 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.
- $Prop(G') = \bigcup_{j=1}^{m} Prop(t_j)$ 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.

4 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 (Blache and Rauzy, 2012). 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.

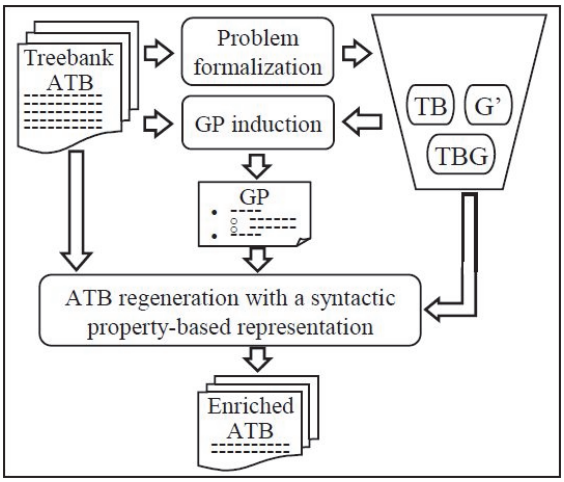

Figure 1: The ATB enrichment method.

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 (Bensalem and Elkarwi, 2014), 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).

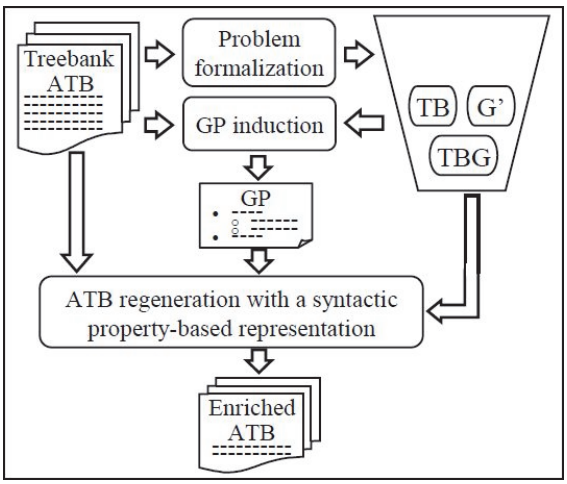

Figure 2: The induction phase of the GP from the ATB.

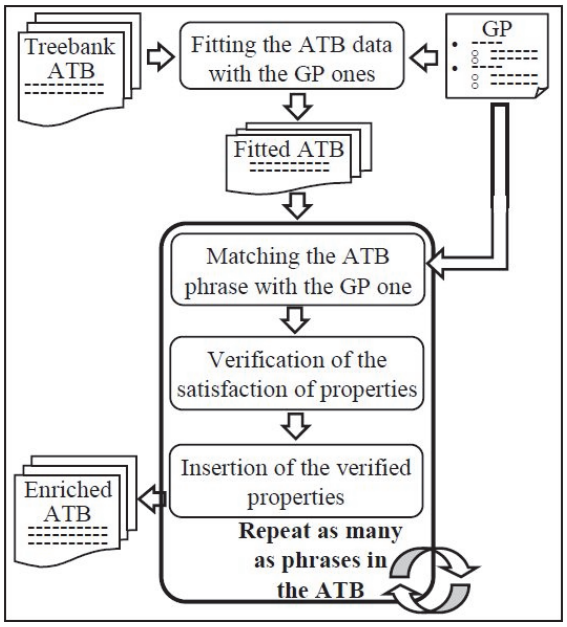

Figure 3: The ATB regeneration phase with a syntactic property-based representation.

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 (Bensalem et al., 2014).

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 in more detail the different steps of this phase.

4.1 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.

4.2 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).

4.3 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.

4.4 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.

5 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 (Blache and Rauzy, 2012). 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 $\mathbf{c_x}$ 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.

5.1 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.

5.2 The Solver of the Linearity Properties

This solver checks the current linearity property of the syntactic category in the GP. The satisfaction is achieved 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.

5.3 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 ca in Const(p), do
  if label(ca)= adjc_p.cx, then
     for each cb in Const(p), do
        if a≠b and label(cb)= adjc_p.cy, 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.

5.4 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 ca in Const(p), do
  if label(ca)= unic_p.cx, 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.

5.5 The Solver of the Obligation Properties

This solver checks the current obligation property of the found GP category. The satisfaction is attained if the obligatory 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 (found=true).

5.6 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(cb)= exig_p.cy, 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).

5.7 The Solver of the Exclusion Properties

This solver checks the current exclusion property of the syntactic category in the GP. The satisfaction is achieved only if the constituents of this property 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.cx, Const(s))
if a > 0, then
  for each cb in Const(s), do
    if a≠b and label(cb)= excl_p.cy, 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).

6 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 are going to present some of the obtained results after citing above the meanings used in the headers of the tables due to lack of space:

Table 1: The header meanings.

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

First, we have found that the ATB is composed of 841 grammatical categories of which 348 are syntactic (put in 21 phrase groups). Table 2 shows the distribution of the ATB phrase by frequency, the possible constituent number and the production rule number.

Table 2: The distribution of the phrases in the ATB.

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

From Table 2, 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 3 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 have 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 an NP if an ADJ (adjective) exists.

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.

Table 3: The distribution of the ATB properties by phrase.

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

For an overview of all the property types, we can observe their high frequency compared to other enriched treebanks (e.g. the FTB) (Blache and Rauzy, 2012). Indeed, according to Table 4, 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 ATB. This reflects the richness and variety of structures in Arabic.

Table 4: The distribution of the properties in treebanks.

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

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.

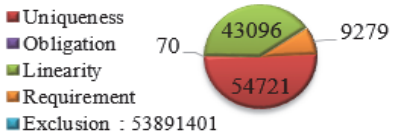

Figure 4: The distribution of the ATB properties by type.

As already mentioned in the previous section, the verification of the property satisfaction distinguishes those satisfied from those violated. We represent in the Table 5 the satisfaction rates of the properties by type of the phrases VP as instance.

Table 5: The satisfaction rates of the VP property types.

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

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. The Table 6 gives us the strong properties of the VP.

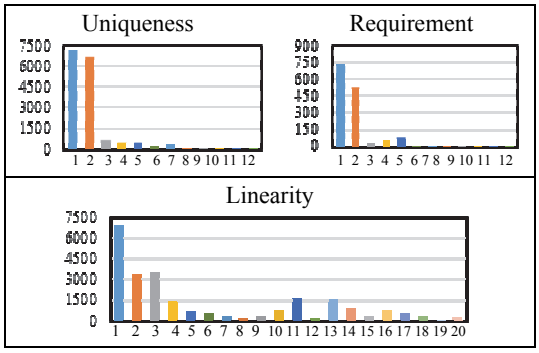

Figure 5: The distribution of the VP properties.

Table 6: The strong properties of the VP.

	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

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.

7 CONCLUSION AND PERSPECTIVES

We have described in the present paper a formalization of the enrichment method, which consists of adding syntactic properties to the existing annotation. This method is based on three main phases: the enrichment problem formalization, the GP induction and the regeneration of the ATB with property annotations. The heart of the enrichment method is specially in the third phase. It consists on the verification of the satisfaction of the GP property for each ATB phrase. The verification result is used to enrich the ATB. We had good experimentation results and various properties of different types in the enriched ATB.

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 modify the description of the obligation and exclusion properties. In future works, we can optimize our enrichment method by integrating several control mechanisms into determining syntactic categories and verification of their properties. We can go further by applying our enrichment method to other annotated corpora obtained from existing parsers.

REFERENCES

Abdul-Mageed, M., Diab, M., 2012. AWATIF: A Multi-Genre Corpus for Modern Standard Arabic Subjectivity and Sentiment Analysis. *Language Resources and Evaluation Conference (LREC'12)*, Istanbul, Turkey.

Alkuhlani, S., Habash, N., 2011. A Corpus for Modeling Morpho-Syntactic Agreement in Arabic: Gender, Number and Rationality. *Association for Computational Linguistics (ACL'11)*, Portland, Oregon, USA.

Alkuhlani, S., Habash, N., Roth, R., 2013. 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.

Bensalem R. B., Elkarwi, M., 2014. 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.

R. B. Bensalem, Elkarwi, M., Haddar, K., Blache, P., 2014. 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.

Blache, P., Rauzy, S., 2012. Hybridization and treebank enrichment with constraint-based representations. *LREC'12- Workshop on Advanced Treebanking*. Istanbul. Turkey.

Cahill, A., 2008. Treebank-Based Probabilistic Phrase Structure Parsing. *Language and Linguistics Compass 2 (1),* 18-40.

Çakıcı, R., 2005. Automatic induction of a CCG grammar for Turkish. *ACL Student Research Workshop*, pp. 73–78, Ann Arbor, Michigan.

El-taher, A. I., Abo Bakr, H. M., Zidan, I., Shaalan, K., 2014. An Arabic CCG approach for determining constituent types from Arabic treebank. *Journal of King Saud University - Computer and Information Sciences*, pp. 1319-1578.

Habash, N., Rambow O., 2005. Arabic Tokenization, Part-of-Speech Tagging and Morphological Disambiguation in One Fell Swoop. *ACL*, pp. 573-580, Ann Arbor, Michigan.

Hovy, E., Marcus, M., Palmer, M., Ramshaw, L., Weischedel, R., 2006. OntoNotes: The 90% Solution. *North American Chapter of the Association for Computational Linguistics (NAACL'06)*, pp. 57–60, USA.

Koller, A., Thater, S., 2010. Computing weakest readings. *ACL*, Uppsala, Sweden.

Maamouri, M., Bies, A., Buckwalter, T., Mekki, W., 2004. The Penn Arabic Treebank: Building a Large-Scale Annotated Arabic Corpus. *NEMLAR Conference on Arabic Language Resources and Tools,* Cairo, Egypt.

Müller, H. H., 2010. 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.

Oepen, S., Flickinger, D., Toutanova, K., Manning, C. D., 2002. LinGO Redwoods - A Rich and Dynamic Treebank for HPSG. *LREC'02 - workshop on parsing evaluation*, Las Palmas, Spain.

Palmer, M., Babko-Malaya, O., Bies, A., Diab, M., Maamouri, M., Mansouri, A., Zaghouani, W., 2008. A Pilot Arabic Propbank. *LREC'08*, Marrakech, Morocco.

Tounsi, L., Attia, M., Van-Genabith, J., 2009. 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.