An Arabic Probabilistic Parser based on a Property Grammar

Raja Bensalem, Kais Haddar, Philippe Blache

An Arabic Probabilistic Parser based on a Property Grammar

Raja BENSALEM

Faculty of Economic Sciences and Management of Sfax, Computer Sciences Department, University of Sfax, TN, raja_ben_salem@yahoo.com

Kais HADDAR

Faculty of Sciences of Sfax, Computer Sciences Department, University of Sfax, TN, kais.haddar@yahoo.fr

Philippe BLACHE

Laboratoire Parole et Langage (LPL), Aix en Provence, Provence-Alpes-Côte d'Azur, FR, blache@lpl-aix.fr

The specificities of the Arabic parsing such as the agglutination, the vocalization and the relatively order-free of words in the Arabic sentences, remain a major issue to consider. To promote its robustness, such parser should define different types of constraints. The Property Grammar formalism (PG) verify the satisfiability of the constraints directly on the units of the structure, thanks to its properties (or relations). In this context, we propose to build a probabilistic parser with syntactic properties, using a PG, and we measure the production rules in terms of different implicit information and in particular the syntactic properties. We experimented our parser on the treebank ATB using the parsing algorithm CYK and obtained encouraging results. Our method is also automatic for the implementation of most property type. Its generalization for other languages or corpus domains (using treebanks) could be a good perspective. Its combination with the pre-trained models of BERT may also make our parser faster.

Keywords: Probabilistic parser, Property Grammar formalism, Arabic language, Lexicalized grammar

1 INTRODUCTION

To promote the robustness of the parser, man should consider different types of constraints. To verify if these constraints are satisfied, the parser should be able to access directly to the variable values of the constraint without the mediating influence of any higher-level structures or elements [Blache 2016a]. The Property Grammar formalism (PG) [Blache 2006] is a good alternative because it represents the constraints as properties (or relations) between the units of a structure directly and independently of their structure and their order. The obtained grammar could be a main component of a statistical parser for the Arabic language.

To build such grammar, we should consider many challenges such as:

- The choice of a suitable learning corpus that deals with the Arabic specificities,
- The complexity of the generated PG that may be exponential due to the type-free and position-free of its units, and also due to the strong granularity of its learning corpus annotations, and
- The interpretation of the properties that could require heuristics or data with different format based on external tools.

This article fits into this context and proposes to build a probabilistic parser with syntactic properties. This parser is based on two main components: the learning model and the analysis algorithm. We build the learning model from the treebank ATB [Maamouri 2008, Kulick 2010], to obtain a lexicalized Probabilistic Context-Free Grammar (PCFG) and a lexicalized Probabilistic Property Grammar (PPG). In these grammars, we measure the production rules in terms of probabilities of different implicit information such as the lexical category sequences, the Head structures and in particular the syntactic properties. We choose the analysis algorithm CYK to experiment our learning model on a given test corpus. We combine it with the Viterbi algorithm to optimize our analysis.

Based on our knowledge, this is the first Arabic parser of this type. The experimentation has led to encouraging results, giving new information to the parses, which are the syntactic properties. This parser may be beneficial in different research areas in the automatic language processing. Our method is also automatic. Its generalization for other languages or corpus domains could be a good perspective.

This paper is organized as follows: Section 2 includes an overview of some Arabic statistical models that learn linguistic dependency constraints and some application domains of the PG formalism. Section 3 presents the property grammar formalism, defines its components and its operating mode and describes the existing Arabic property grammar. Section 4 explains the proposed parsing method and shows a set of preprocessing functions of the entries. Section 5 shows the experiments and an evaluation of our parser compared to the state-of-the-art Stanford parser. Section 6 talks about open problems including the specificities of the Arabic language that could generate different ambiguities. Section 7 tackles the conclusion and some perspectives.

2375-4699/2023/MonthOfPublication - ArticleNumber $15.00

http://dx.doi.org/10.1145/3612921

2 RELATED WORK

There are two work areas to observe in the following subsections: Arabic statistical models that integrate linguistic dependency constraints and the property grammar works.

2.1 Arabic statistical models

Statistical models provides two representation forms of the training data: the classical classifiers of word embeddings (dense vectors) and the pre-trained models.

2.1.1 Classical classifiers

In [Abdelrazaq 2018, Aqel 2019], the researchers propose an Inductive Learning Algorithm (ILA) to produce a set of classified parsing rules for parsing Arabic nominal and verbal sentences. This contribution generates good accuracy results especially for verbal sentences.

In 2017, Soliman et al. presented the open-source project AraVec that builds a pre-trained distributed word embedding representation based on the continuous bag-of-words (CBOW) and Skip-Gram techniques [Soliman 2017], and uses the K-means algorithm for the vector clustering. Its newest version AraVec 3.0[1] provides 16 different word embedding models (using more than 3,300,000,000 tokens) using the Python library, Gensim. These models are built on two different Arabic content domains; varied dialectal Tweets and Wikipedia Arabic articles. AraVec 3.0 can also produce two types of models: unigrams and n-grams models, which were not possible in the previous versions..

The Stanford parser is also in evolution. It provided a fast transition-based dependency representation by training a compact Neural Networks classifier, which accepts word embedding inputs. This classifier uses only 200 learned dense features in place of the sparse indicator features [Chen-Manning 2014]. Using this technique, the Stanford team generates in 2016 a multilingual treebank collection for which, it trains existing treebanks for 33 languages including the Arabic [Nivre 2016] and wins in the CoNLL 2017 Shared Task [Dozat 2017]. The newest version of the parsing models 4.2.0[2] is provided in the end of 2020. In 2022, Saber et al. [Saber 2022] used Stanford parser and POS-tagger to generate a rule-based model for automatic ontology extraction. The latter is applied to extract the triple attributes of a sentence (subject, predicate, and object) from the parsing tree.

ArWordVec [Fouad 2020] built its word embedding models on large Arabic tweets datasets, using three different approaches: CBOW, Skip-Gram and global vectors (GloVe). It uses a variant of the algorithm SVM or the Naives Bayes for the vector clustering.

Maalej et al. [Maalej 2021] used for the Arabic parsing, three deep learning techniques: LSTM, GRU and BI-LSTM. They applied the bag of words (BOW) representation for word embeddings. They created a model for each syntactic level in the ATB to determinate constituents of a sentence and relations between them.

Morad et al. [Morad 2021] combines both deep learning model and dense Arabic word representations to generate constituent parse tree in a complete workflow.

2.1.2 Pre-trained Models

The word embedding representations have only one layer of initialized weights on their parameters, which could be useful to capture semantic meanings of the words. The pre-trained model representations employ many layers to define the best weights. This could be useful to define a higher parsing level (than the semantic meanings) such as anaphora, long-term dependencies, agreement, and negation. There are some works for the Arabic that use famous pre-trained models as follows:

Che et al. [Che 2018] propose in 2018 the multilingual parser HIT-SCIR that combines the Stanford's winning parser for the CoNLL 2017 [Dozat 2017] with deep contextualized word Embeddings from Language Models (ELMo) [Peters 2018, Che 2018, Zeman 2018].

BERT [Devlin 2019] is a pre-trained model proposed by Google and used by many parsers including the Arabic ones. It stands for Bidirectional Encoder Representations from Transformers. In contrast to the contributions of ELMo and HITSCIR [Peters 2018, Che 2018], which use a shallow concatenation of independently trained left-to-right and right-to-left LMs, BERT pre-trains deep bidirectional representations from unlabeled text by applying joint conditioning parameters on both left and right context in all 12 layers.. For the Arabic, there are many BERT-based parsers, for example: the last Berkeley parser [Kitaev 2019], Arabic BERT [Safaya 2020] and AraBERT [Antoun 2021]. In Appendix A, we give more details about these BERT-based parsers.

[1] https://github.com/bakrianoo/aravec
[2] https://nlp.stanford.edu/software/lex-parser.shtml

Recently, AraBERTv2 model provides the best scores for three Arabic treebanks (PADT, CATiB and ArPoT) [Al-Ghamdi 2023].

ULMFiT (Universal Language Model Fine-tuning for Text classification) [Howard 2018] is an open-source pre-trained model learned on Wikipedia articles. Its Arabic version[3] was implemented by Muhammad Khalifa in 2019. . MultiFiT was also implemented for other languages [Eisenschlos 2019] and it significantly outperformed the zero-shot model LASER [Artetxe 2019] and multi-lingual BERT [Devlin 2019].

The summary table 1 shows a comparison between the described Arabic statistical parsers.

Table 1: Comparative table of parsers

Parsers	Learning algorithm	Used technique	Test corpus	F-mesure
Abu-Soud et al. [Abusoud 2018]	ILA	Classification	376 sentences	92,63
Maalej et al. [Maalej 2021]	LSTM, GRU, BI-LSTM	BOW	PATB v3.2 (402,291 words)	99,60
AraVec [Soliman 2017]	K-Means	CBOW and Skip-Gram	STS wiki (250 pairs of sentences)	74,40
Stanford [Nivre 2020]	Neural Networks	Grammatical dependencies	PADT (CoNLL 2017 shared task)	80,67
Saber et al. [Saber 2022]				73,60
ArWordVec [Fouad 2020]	SVM or naïve Bayes	CBOW, Skip-Gram and GloVe	Arabic Tweets datasets	74,15
HIT-SCIR (ELMo+Stanford) [Che 2018]	LSTM	Deep contextualized word Embeddings	PADT (CoNLL 2018 shared task)	83,00
Berkeley [Kitaev 2019]	CNN	Bidirectional Representations (BERT)	SPMRL 2013/2014 shared task datasets	-
Arabic BERT [Safaya 2020]	CNN	Bidirectional Representations (BERT)	2000 tweets	89,70
AraBERT [Antoun 2021]	CNN	Bidirectional Representations (BERT)	Sentiment analysis (HARD, ASTD, …), NER (ANERcorp), Question Answering (ARCD)	84,20
Al-Ghamdi et al. [Al-Ghamdi 2023]	CNN	AraBERT[Antoun 2021] + Fine-Tuning	PADT, CATiB and ArPoT (CATiB)	87,65
ULMFiT for Arabic (2019)	RNN	discriminative fine-tuning, STLR and gradual unfreezing	IMDb, TREC-6 and AG datasets	85,75

2.2 Property grammars

Several works benefited from the PG formalism [Blache 2006] in different search areas. In the context of symbolic syntactic analysis, Duchier et al. [Duchier 2010] presents a formal semantic definition of the PG. They apply this definition to model PG parsing as a constraint satisfaction problem. In 2012, Duchier et al. [Duchier 2012] propose an extension from this model to process new property types. This extension transforms the syntactic relations on feature structures.

Later, Blache built a Chinese constraint grammar from the Chinese treebank CTB [Blache 2014] and Bensalem and Elkaroui [Bensalem 2014a] induced a variable granularity PG from the Arabic treebank ATB.

Using the PG formalism, Blache and Rauzy [Blache 2012] propose, an approach of hybridization of a symbolic control and a probabilistic parsing. This approach is based on heuristics (weights) that are calculated on the occurrences of the satisfied properties.

This formalism was also used in the enrichment of the treebanks: FTB for the French [Rauzy 2012] and the ATB for the Arabic [Bensalem 2016]. The latter authors enriched the ATB with syntactic properties thanks to a formal modeling method [Bensalem 2015a]. Bensalem et al. [Bensalem 2018] also applied this PG enrichment to the parsing results of the Stanford Parser and evaluated it in terms of the satisfied properties.

Blache et al. [Blache 2016b] presented in 2016 Marsa-Gram, a multi-lingual system that provides a method of inference of properties from different treebanks based on property metrics. They applied this method to the Universal Dependencies Treebank to reconstitute language families using the hierarchical clustering.

[3] https://github.com/mukhal/arabic-UlmFit

The PGs was also useful to generate the grammars of under-resourced languages such as the Yoruba. According to the formalism of Hybrid Womb Grammars, Adebara [Adebara 2019] generates a Yoruba grammar by inducing a PG from an English CFG and including then a Yoruba lexicon and representative Yoruba input phrases.

3 PG FORMALISM

The PG formalism proposes a constraint-based approach [Blache 2006]. It provides a local and decentralized representation of linguistic information. It is a phrase grammar because it is organized as hierarchical syntactic structure. Thus, a PG describes each category with as a set of properties that express different relations between the units of a syntactic structure.

3.1 Properties

The properties are all defined at the same level, i.e. they are neither inter-dependent, nor inter-ordered. These properties can be on lexical level (such as morphological or phonological properties) or on syntactic level. The syntactic properties are of six types as shown in the following Table 2:

Table 2: Property Functions in the GP

Properties	Functions
Constituency (Const)	Set of constituents in the syntactic structure S
Uniqueness (Unic)	Set of constituents that appear only once in S
Obligation (Oblig)	Set of obligatory constituents (Heads) in S
Linearity ($\prec$)	Linear Precedence between constituents in S
Requirement ($\Rightarrow$)	Obligation of co-occurrence between constituents in S
Exclusion ($\otimes$)	Restriction of co-occurrence between constituents in S

The properties of constituency, uniqueness and obligation are unary relations. The others are binary ones.

3.2 Verification of the constraint satisfaction

Each syntactic category has a set of constraint subsystems per property type. The parsing using this formalism refers to verify for each syntactic category the satisfiability of its subsystems. To parse a given statement, a three-step process must be applied: The enumeration of all the possible categories of this statement, the construction of possible sequences of enumerated categories, and finally the direct verification of the consistency of the constraint subsystems to the syntactic categories of these sequences.

3.3 Arabic PG

According to [Bensalem 2014a], the Arabic PG is not directly induced from the treebank. It needs to be induced from a CFG (Context-Free Grammar) that is induced from the treebank. The CFG is a set of production rules. To generate them, it applies recursively an in-depth parse of the parse trees of the treebank ATB. The PG extract its properties from the obtained CFG. Thus, it describes for each syntactic category various types of syntactic properties. It applies for each property a specific algorithmic interpretation [Bensalem 2014a].

4 PROPOSED APPROACH FOR THE PROBABILISTIC PROPERTY-BASED PARSING

As shown in Figure 1, there are two main components in our parser: the learning model and the parsing algorithm. We train our supervised model on a PCFG induced from a Treebank. The parsing decisions depends not only on the production rules of the PCFG but also on different property types. We obtain these properties from a PPG (Probabilistic PG). To summarize, to build the learning model, we need to induce a PCFG from a learning corpus (a treebank), and to induce syntactic properties from the production rules of the PCFG. The parsing algorithm is used after that to generate a parse tree for a given lexically processed sentence.

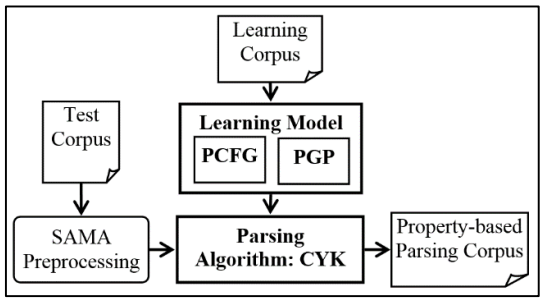


Figure 1: General architecture of our property-based parser

In the architecture described in Figure 1, we chose the parsing algorithm CYK [Kasami 1965, Younger 1967]. We combined to it the Viterbi optimization algorithm. We preprocessed the test corpus with the SAMA tool [Maamouri 2008] to provide their morphological analysis and POS tagging before its parsing with the CYK algorithm.

4.1 Learning model building

To obtain the PCFG and the PPG of our learning model we should pre-process our learning corpus, induce the PCFG from it and the PPG from the PCFG. We explained more these steps in the following sub-sections:

4.1.1 Pre-processing of the learning corpus

The treebank ATB, which is our learning corpus, includes for each sentence its hierarchical structure and morpho-syntactic POS-tags (categories) for the words. To induce a deep PCFG from the treebank, we specified for each syntactic category, the lexical categories of its constituents.

In addition, since we cannot generate explicitly the obligation properties (or heads) of the syntactic structure, we apply the syntactic rules of Habash et al. [Habash 2009] for each line in the corpus similarly to the lexicalization process. We specified, before, the indexes of the first and the last word of each syntactic structure. Habash et al. apply syntactic rules on the sentence and affect to the head of each structure the smallest index.

We control also the case of empty lists (tagged as LST whose size is 1).

The treebank words in the tree format have transliterated writing. The words of the test corpus have, by contrast, Arabic writing. We need to have also the Arabic writing in the learning corpus for their matching with the test corpus words, using the format "pos" provided in the treebank. This format presents data as a set of fields that specifies a token in the syntactic tree. The Arabic writing of the token is one of these fields.

4.1.2 Induction of the deep lexicalized PCFG as CNF

Our PCFG to induce is special because it is lexicalized, deep and has a Chomsky Normal Form (CNF) (Chomsky, 1959). To obtain it, we followed the method shown in Figure 2 :

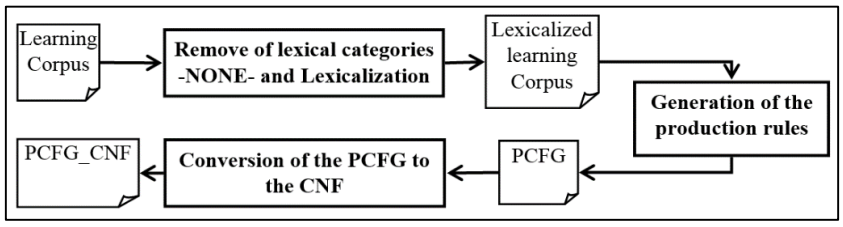


Figure 2: The induction method of the PCFG-CNF

As shown in Figure 2, we remove first the lexical categories -NONE- from the learning corpus because they describe empty words. In the lexicalization step, we associate to each syntactic category its Head category, and the sequence of the lexical categories of its constituents. The following example shows the syntactic structure of an Arabic sentence taken from the ATB [Maamouri 2008] before and after lexicalization step and its generated production rules (which belong to the PCFG):

Sentence	كتب وليد شقير يقول:	ktb wlyd $qyr yqwl :	Walid Chequir wrote :
Syntactic structure before lexicalization	**(S** **(VP** (PV+PVSUFF_SUBJ:3MS ktb) **(NP-SBJ-1** (NOUN_PROP wlyd) (NOUN_PROP $qyr)) **(FRAG** **(VP** (IV3MS+IV+IVSUFF_MOOD:I yqwl) **(NP-SBJ-1** (-NONE- *)))) (PUNC :)))		
Syntactic structure after lexicalization	**(S(VP)** **(VP(PV+PVSUFF_SUBJ:3MS)** (PV+PVSUFF_SUBJ:3MS ktb) **(NP-SBJ-1(NOUN_PROP)** (NOUN_PROP wlyd) (NOUN_PROP $qyr)) **(FRAG(VP)** **(VP(IV3MS+IV+IVSUFF_MOOD:I)** (IV3MS+IV+IVSUFF_MOOD:I yqwl)) (PUNC :)))		
Production rules	S(VP) → VP(PV+PVSUFF_SUBJ:3MS) PUNC VP(PV+PVSUFF_SUBJ:3MS) → PV+PVSUFF_SUBJ:3MS NP-SBJ-1(NOUN_PROP) FRAG(VP) NP-SBJ-1(NOUN_PROP) → NOUN_PROP NOUN_PROP FRAG(VP) → VP(IV3MS+IV+IVSUFF_MOOD:I) VP(IV3MS+IV+IVSUFF_MOOD:I) → IV3MS+IV+IVSUFF_MOOD:I PV+PVSUFF_SUBJ:3MS → ktb NOUN_PROP → wlyd NOUN_PROP → $qyr IV3MS+IV+IVSUFF_MOOD:I → yqwl PUNC → :		

To make the obtained PCFG useful by the algorithm CYK, we should convert it to the CYK. Thus, we transform all rules of the PCFG into only two possible forms: rules with single rhs(right-hand side) as X → w or rules with binary rhs as X → Y Z (with X, Y and Z are syntactic or lexical categories, and w is a word of the corpus). We obtain a PCFG in the CNF form, called PCFG-CNF. For that, we apply the following steps:

1. Copy the rules of the PCFG that have the CNF form in PCFG-CNF. This is the case of:
   - The rules whose lhs(left-hand side) is a lexical category and rhs is a word,
   - The rules whose rhs is a sequence of two lexical categories.
2. Reformulate or ignore the other rules in the PCFG-CNF according to the following cases:
   - The rules whose rhs is only one category: to ignore.
   - The rules whose rhs includes more than one category, and at least one of the categories is syntactic. For this case, we should first scan the categories of this rhs to replace any syntactic category that includes one lexical category by the latter.

When the rhs of a rule includes more than two categories, we should add a specific processing that transforms this rule into a series of rules that respect the CNF. In fact, we consider that each rule has the form X → Y Z such that X refers to the category Z of the preceding rule. The algorithm 1 shows this transformation for a given rule. It transforms an original rule X → C1 C2 C3 C4 (where its C1, C2, C3 and C4 are grammatical categories) first to X → C1 X(C1) to represent the first category in the rhs of the original rule. It adds other attached rules to represent the other categories. The number of the added rules in PCFG-CNF is equal to the number of categories in the rhs of the original rule -1. We sort these rules by category in the lhs. We calculate also the probability of each one according to their occurrence number.

ALGORITHM 1: Processing of the rules whose rhs includes more than two categories

```
rs :Rule List
Begin
   r : Rule , h: integer
   n ← 3
   while n ≤ maxL
      for r is in rs
         if length(r.rhs) = n then
            r.rhs(2) ← r.lhs +"("+r.rhs(1)+")"
            r.addAttached(r.lhs, r.rhs)                    //to add specific rules
         end if
         n ← n+1
      end for
   end while
End
```

4.1.3 Induction of the PPG lexicalized as CNF

The PPG is directly induced from the lexicalized PCFG since the consistency of its hierarchical representation to the PG formalism. The PPG is a grammar that defines a set of relations between grammatical categories not in terms of production rules (like the PCFG) but in terms of local constraints (so-called properties). We restrict our PPG to syntactic properties, as indicated in Table 2, and particularly, the constituency (const), the obligation (oblig) and the uniqueness (unic) the linear precedence ($\prec$), the mandatory co-occurrence ($\Rightarrow$), the restricted co-occurrence ($\otimes$). The output of this step is a lexicalized PPG under the CNF (denoted PPG-CNF).

This grammar can be defined by a 4-tuple PPG= {N, Σ, P, Vp} where:

- N is a finite set of lexicalized syntactic categories (like dPCFG),
- Σ is a finite set of lexical categories (like dPCFG),
- P is a finite set of syntactic properties that links ∀ α ∈ N, β1 et β2 ∈ (N U ΣN in any of the following 6 ways: βn ∈ const(α), βα ∈ olig(α), βα ∈ unic(α), α : β1$\prec$ β2, α : β1$\Rightarrow$ β2 and α : β1$\otimes$ β2. We deduce each of these properties from the rule set R defined in dPCFG.
- Vp is the set of the probabilities associated to the properties that describe each lexicalized syntactic category in N. The probability of a property propcti ∈ P (where t is its type and c is the syntactic category that describe) is equal to:

$$P(prop_{cti}) = \frac{occ(prop_{cti})}{\sum_{c \in C} \sum_{i \in I} occ(prop_{cti})} \qquad (1)$$

To induce the set of properties that describe each syntactic category c, we specify each time a property and we verify if it is correct for all the production rules whose lhs is equal to c. A property is interpreted differently from one type to another as follows:

- Constituency, which defines, for each lexicalized syntactic category c(h), a set of constituency properties const(c(h)). This is the set of the grammatical categories that participate to construct the set of the Right-Hand-Sides of rules of c(h) : RHS(c(h)).
- Obligation, which specifies for each syntactic category c(h), the set of the heads of a syntactic structure. This could be obtained by means of external specific rule sets (such as the rules of [Habash, 2010].
- Uniqueness, which provides the set of grammatical categories that cannot be repeated in any rhs of the related rules.

```
∀ rhsk ∈ RHS(c(h))
    ∀ (ci, cj) ∈ rhsk
    if ci ≠ cj then
        add uniq(c(h), ci)
```

- Linearity, which verifies the linear precedence relations between the constituents of the lexicalized syntactic category c(h). If a relation if valid in all the RHS(c(h)), it will be added to the linearity set.

$\forall$ $rhs_k \in RHS(c(h))$
  if $\exists$ $(c_i, c_j) \in rhs_k \mid c_i \prec c_j$ **and** $\nexists$ $(c_i, c_j) \in rhs_k \mid c_j \prec c_i$ **then**
    add lin(c(h), (c_i , c_i))

- Requirement, which identifies the set of pairs of categories such that the appearance of the first requires the appearance of the second in the same rhs.

$\forall$ $rhs_k \in RHS(c(h))$
  verif $\leftarrow c_i \in rhs_k \wedge c_j \in rhs_k$
if verif **then**
  add req(c(h), (c_i , c_i))

- Exclusion, which determines the set of pairs of categories that never co-occur in the same rhs. However, this automatic interpretation could leads to an over-generation of the number of such properties.

$\forall$ $rhs_k \in RHS(c(h))$
  verif $\leftarrow \neg (c_i \in rhs_k \wedge c_j \in rhs_k)$
if verif **then**
  add excl(c(h), (c_i , c_i))

The following interpretations are inspired from the contribution of [Blache and Rauzy, 2012].

4.1.4 Learning model generation

The learning model output is a hybridization of PCFG production rules with the PPG properties. That is to verify the satisfaction of each property $prop_{cti} \in P$ in each production rule r_k, whose lhs is the category c(h) that describes $prop_{cti} \in P$. Formally, we can model this problem by the 8-tuple = (N, R, D, Vd, Const(R), Const(P), P', PF) where:

- N is a finite set of lexicalized syntactic categories (like dPCFG and PPG).
- R is a finite set of production rules (like dPCFG)
- D is a set of the possible sequences of the lexical categories that appear directly or indirectly in each syntactic structure,
- Vd is the set of probabilities of the lexical category sequences of D. Such probability is useful particularly in the parsing task to optimize the choice of rules.
- Const(R)= $\bigcup_{k\in K} Const(r_k)$: is the set of the constituents of the RHS in each production rule of R,
- Const(P) = $\bigcup_{t\in T} \bigcup_{i\in I} Const(prop_{cti})$: is the set of the constituents linked in each syntactic property $prop_{cti}$ of P,
- P' is the set of satisfied properties $prop_{cti} \in P$ in each rule $r_k \in R$ where $c(h) \in N$, $\mathrm{lhs}(r_k) = \mathrm{c(h)}$, and $\mathrm{Const}(prop_{cti}) \in \mathrm{Const}(r_k)$. Thus, a property is satisfied in a rule if its constituents appear in the rhs of this rule.
- PF is the set of probability functions fp(r_k) of the satisfied properties of P' associated each one to a rule r_k. It is calculated as follows:

$$fp(r_k) = \sum_{t\in T} \sum_{i\in I} w_t \times P(prop_{cti}) \qquad (2)$$

Where $\mathrm{lhs}(r_k) = \mathrm{c(h)}$, $\mathrm{Const}(prop_{cti}) \in \mathrm{Const}(r_k)$ and w_t is the weight of the properties of the type t. Its numerical value is chosen by the user or estimated using an adjustment function.

In the step of parsing, the probability to choose a rule r_k depends not only on the value of fp(r_k) but also on the probabilities.

4.2 Application of the parsing algorithm: CYK

We chose the parsing algorithm CYK whose the following qualities satisfy our parsing needs:

- Speed of Processing: this algorithm loads all the rules in a table for once, unlike Earley's algorithm [Earley 1970] that only loads the rules describing the needed categories at the current step.
- Insensitivity to recursive rules: in the recursive rule, the lhs is one of the constituents of the rhs.
- Simplicity of browsing: this algorithm uses sequentially a list of rules and not apply the operations of the Earley algorithm (reading, prediction and completion).

We chose the CYK algorithm also because we observed that it is more spread for many parsing tasks [Zanzotto 2020, Khatun 2021, Chen 2021, Sahay 2021, Dordevic 2020, Zhang 2020, Paranjpe 2021, Li 2020].

Before applying the CYK algorithm, we should pre-process the test corpus and recognize the various constituents of their lines.

4.2.1 Pre-processing of the test data

The preprocessing of the test corpus includes the morphological parsing and the morphological disambiguation. Several tools[4] can do that. The SAMA tool prove high performance (accuracy and speed) especially for the Standard Arabic. It is used to build the PATB morphological analyses and it shows its analyzing capacity in many works [Mahyoob 2020, Inoue 2021, Khalifa 2020, Taji 2018].

We used the SAMA implicitly. Indeed, we extracted from the reference corpus the correct lexical categories of the words after disambiguation (using the tool MADA (Morphological Analysis and Disambiguation for Arabic) [Habash 2013]) and used it as the input corpora in the test phase.

4.2.2 Extraction of the constituents

Each sentence in our corpus test contains segmented and lexically annotated words. We define first these words and their lexical categories to search later their lexical rules in the PCFG-CNF. For that, we apply a nested division that decomposes each line (sentence) into word-category pairs (noted < w, c >), then each pair into its two constituents.

4.2.3 Parsing using the CYK algorithm

To apply the CYK algorithm we need to fill a specific chart for each test sentence. This chart includes all possible PCFG rules to use in a bottom-up strategy. The processing starts incrementally from the single word to the sequence of all words in the sentence. We fill the chart at the first level with lexical rules $X \rightarrow w$, and at the following levels, with syntactic rules $X \rightarrow Y\ Z$. The syntactic rules cover a sequence $S_{i,j}=(w_i,c_i)\ldots(w_j, c_j)$ where $1 \leq i < j \leq |S|$. We fix for that a separator $i \leq k < j$ that define the sub-sequences $S_{i,k}$ and $S_{k+1,j}$ to cover respectively by the rhs components in each loaded rule.

To select the best collection of rules that cover the test sentence, we use the Viterbi algorithm as optimization algorithm We found that Viterbi gives good results for parsing in many works ([Hayashi 2017, Nishida 2020]). This algorithm helps to choose at each step and for each syntactic category the best parse. This allows us to benefit from both the speed of the local search algorithm and avoid its default, which is the risk of falling into a local optimum. The choice decision of the rule $X \rightarrow Y\ Z$ noted d_a that covers the sequence of words $(w_i..w_j)$ depends on the conditional probability of this rule given the best probabilities of the rules previously chosen, where:

$$\begin{aligned} P(d_a|d_1\ d_2\ \ldots\ d_{a-1}) &= P(d_a) \times P(d_{a-1}|d_1\ d_2\ \ldots\ d_{a-2}) \\ &= P_{[i,j]}(X \rightarrow YZ) \times \max(\mathrm{Y}, \mathrm{i}, \mathrm{k}) \times \max(\mathrm{Z}, \mathrm{k}+1, \mathrm{j}) \end{aligned} \quad (3)$$

The probability of the rule depends not only on the probability function fp(d_a) but also on a compliance parameter pc:

$$P(d_a) = fp(d_a) + pc \quad \text{Where} \quad pc = \begin{cases} \dfrac{\mathrm{occ}(\mathrm{sequence}_s(X \rightarrow YZ))}{\mathrm{occ}(X \rightarrow YZ)} & \text{if } (c_i \ldots c_j) = \mathrm{sequence}_s(X \rightarrow YZ) \\ 0 & \text{Otherwise} \end{cases} \quad (4)$$

As the formula above shows, the parameter pc is positive if the sequence of lexical categories of the test line $(c_i..c_j)$, is identical to the sequence of lexical categories that describes the chosen rule.

[4] Tools like BAMA, SAMA, MADAMIRA and Alkhalil MorphoSys2

5 EVALUATION AND EXPERIMENTATION

The experimentation of our parsing approach on an input corpus and a test corpus gives us several results that we show in the following subsections. Before that, we present the meanings of the symbols in the list headings in Table 3:

Table 3: Meanings of the symbols in the table headings

Symbols	Meanings	Symbols	Meanings
XP	Syntactic category	#C	Cardinal of the possible constituents
#	Occurrence number	#R	Cardinal of the rules
∑	Total	#P	Cardinal of properties
Const	Set of constituency properties	Lin	Set of linearity properties
Uniq	Set of uniqueness properties	Req	Set of requirement properties
Oblig	Set of obligation properties	Excl	Set of Exclusion properties

5.1 Characteristics of the inputs

To run our probabilistic property parser, we need to use the following three entries: the learning corpus, the test corpus, and the rules of [Habash 2009]. We run the parser on a Core i3 with 1,70 Ghz of CPU and 4 Go of RAM.

5.1.1 Learning Corpus: the ATB treebank

We tested our parser on the ATB2v1 [Maamouri 2008], which is composed of 501 press articles, including 144,199 segments with high granularity level and very varied syntactic and lexical categories, which promotes its robustness. Its grammar is adapted to the MSA, the input data we are studying. In addition, only the ATB offers a constituency-based representation for its syntactic structures [Bensalem 2014; Al-Ghamdi 2021; Habash 2022]. This representation is adequate with the Property Grammar formalism.

5.1.2 Test corpus: Treebank ATB

Our test corpus is an excerpt from the ATB, in which we assign lexical categories to its words using the "pos" format. It contains 400 sentences of different lengths (or numbers of words). To analyze this corpus with the CYK algorithm, we propose the following equally division according to the sentence lengths. The following table 4 gives the frequencies of the sentences, lexical categories and words in the ATB and in our excerpt to show its coverage level.

Table 4: Frequencies of the sentences, the words and the lexical categories of our excerpt compared to the ATB

Length	ATB						Our excerpt			
	Sentences		Lexical categories		Words		Lexical categories		Words	
	#	%	#	%	#	%	#	%	#	%
[1, 10]	2031	26%	4086	33%	266	2%	407	66%	106	17%
[11, 20]	1562	20%	5800	24%	298	1%	777	51%	134	9%
[21, 30]	1245	16%	6748	21%	291	1%	1182	46%	150	6%
[31, 40]	986	13%	7032	20%	302	1%	1517	43%	168	5%
[41, 50]	749	10%	6937	21%	288	1%	-	-	-	-
[51, 60]	480	6%	5978	23%	281	1%	-	-	-	-
[61, 70]	296	4%	4875	25%	257	1%	-	-	-	-
[71, +∞]	423	5%	7177	19%	302	1%	-	-	-	-
Σ	7772	100%	220080				8252			

In table 4, the column "Sentences" gives for each division the percentage of sentences (under the symbol %) relative to the total number of sentences in the ATB corpus. The column "Words" gives also for each division the number of distinct Arabic words (under the symbol #), and the percentage (under the symbol %) of these words in the entire ATB corpus. For example, the first division [1 , 10] contains 266 distinct words and this represents 2% of the

words in the entire ATB. For our excerpt, the symbol "%" is the percentage of distinct words relative to the occurrence number of words in that excerpt. For example, our excerpt groups in its division [1 , 10], 407 distinct words which represent a percentage of 66% of the words with duplication.

According to Table 4, the number of words in most ATB sentences (75%) does not exceed 40. The most of varied words or categories is grouped in the four first division. This justifies the large coverage of the sentences and their lexical categories and Arabic words that we chose in the excerpt (For example, the coverage of the words is by up to 8 times compared to that of the ATB for divisions [1, 10] and [11, 20].

5.2 Characteristics of the built learning model

Our learning model is the deep lexicalized PCFG-CNF and the lexicalized PPG-CNF.

5.2.1 Deep lexicalized PCFG

Thanks to the lexicalization step, the obtained PCFG is different to that original [Bensalem 2014a], as shown in the table 5. We mention that the symbol # is the occurrence number of distinct syntactic categories, and the symbol #* is the occurrence number of the most frequent syntactic category.

Table 5: Frequencies of the lexicalized rules by syntactic category in the PCFG

XP	#R		#	#*	XP	#R		#	#*
	Before	After				Before	After		
NP	4824	10650	110748	6039	WHADVP	17	26	136	21
PP	263	817	22100	3417	UCP	88	113	132	4
S	1230	4255	19358	1587	SBARQ	51	66	68	2
VP	6675	8683	15947	212	PRN	20	42	65	7
SBAR	380	1512	9524	407	LST	2	2	56	49
WHNP	22	42	4574	571	SQ	26	40	51	4
ADJP	593	759	3665	303	CONJP	2	2	37	34
PRT	14	14	2292	1051	INTJ	1	1	10	10
ADVP	5	11	539	162	X	5	5	5	1
NAC	53	77	221	26	WHPP	3	3	3	1
FRAG	56	64	178	88	∑	14452	27184	1749125	

From Table 5, we can notice that the number of production rules for most syntactic categories has significantly doubled, tripled or even quadrupled after lexicalization. This shows that the composition of most of syntactic category is varied. The less frequent categories have not changed because their rules only appear once as for X and WHPP, or their rules are very few as for INTJ, X, LST and PRT.

This table gives us another information level which is the distribution of the most frequent rule in its syntactic category. For example, the most frequent rule in the category SBARQ appears only twice in the ATB, while there are 65 other rules that cover the rest of the occurrences.

It is also possible to measure the dispersion degree (σ) of the syntactic structures given the frequency of their sequences of lexical categories.

After the conversion of the PCFG to the CNF, the only syntactic categories that not increased their rules are WHPP, because all its right-hand-sides (rhs) contain two categories, and the categories INTJ, PRT and WHADVP, that have a lexical category in their rhs. However, the other syntactic categories increased their rule number up to 8 times.

5.2.2 Lexicalized PPG as CNF

We induced the PPG from the Lexicalized PCFG in the CNF. The table 6 illustrates the frequencies of the obtained properties by syntactic category and property type.

Table 6: Frequencies of the syntactic properties by syntactic category and by type (in thousands)

XP	Const		Uniq		Oblig		Lin		Req		Excl		∑	
NP	2,4	17,6	2,1	17,1	1,6	1,6	2,5	12,9	1,3	15,0	115,1	205,5	124,8	269,7
PP	0,2	1,3	0,2	1,3	0,1	0,1	0,2	1,0	0,2	1,2	1,0	39,1	2,0	44,1
S	0,3	11,6	0,2	11,4	0,1	0,3	0,7	7,0	0,3	10,5	17,7	135,9	19,3	176,7

[5] Occurrence number of categories after remove those that include the constituents -NONE-

VP	0,4	15,8	0,4	15,7	0,2	0,1	3,9	11,9	0,5	13,6	90,1	427,0	95,6	484,2
SBAR	0,4	2,1	0,3	2,1	0,3	0,3	0,3	1,6	0,4	1,9	3,3	18,7	5,0	26,7
WHNP	0,1	0,0	0,1	0,0	0,1	0,0	0,0	0,0	0,0	0,0	0,1	0,0	0,3	0,0
ADJP	0,2	1,3	0,1	1,2	0,1	0,1	0,4	0,9	0,1	1,1	5,8	3,7	6,7	8,2
PRT	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,1	0,0	0,1	0,0
ADVP	0,1	0,0	0,1	0,0	0,1	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,2	0,0
UCP	0,1	11,9	0,1	11,8	0,0	0,3	0,1	7,2	0,1	10,8	0,2	136,0	0,5	178,1
∑	4,3	62,8	3,8	61,9	2,6	3,0	8,3	43,1	3,0	55,2	234,6	966,4	256,6	1192,3

According to Table 6, the number of the properties that describe the syntactic categories has largely increased for most cases. This amounts to an increase in the number of the non-original categories (created to convert the rules to the CNF) such as the categories VP, S, PP and UCP. There is no change for the categories having other non-original ones, such as the categories WHPP and LST.

In terms of types, the obligation properties are the least affected because of their near-absence in the description of the non-original categories, where the head is, in most cases, the first constituent in the rhs of the rules. In addition, the share of the properties of constituency, uniqueness and requirement increased in favor of the exclusion properties. The interpretation of the latter determines the cases where two constituents of a syntactic category never appear together in the rules it describes. In this case, the higher the number of constituents, the more non-appearances of the pairs of these constituents, whereas the non-original categories have few constituents.

5.3 Characteristics of the parsing results using the algorithm CYK

The parsing algorithm is the second component of our property parser. It performs a strategy of ascending analysis of each sentence in the test corpus. Our parser includes not only the tree to the syntactic structures but also their syntactic properties. To evaluate the generated parsing trees, we applied a set of experiments on our parser, its property weights and on the state-of-the-art parser, Stanford Parser. We compared our results to this parser. The next two sub-sections present respectively these different experiments.

For parse trees, we used the metric ParsEval calculated by Evalb[6] tool. Results are automatically generated. For parse properties, we applied the following formulas for all phrases compared to the reference:

$$Precision = \frac{Occurrence\ number\ of\ True\ Found\ Properties}{Occurrence\ number\ of\ Found\ Properties}$$

$$Recall = \frac{Occurrence\ number\ of\ True\ Found\ Properties}{Occurrence\ number\ of\ True\ Properties}$$

$$F - measure = \frac{2 \times Precison \times Recall}{Precison + Recall}$$

5.3.1 Experiments on a state-of-the-art parser: Stanford Parser

The experiments of the Stanford parser on the test corpus provide tree parses with PTB POS-tagging. We adjust the Arabic words of the Stanford parses to their ATB transliterated writings and the ATB tagging to that of the Stanford Parser (the PTB tagging). The ATB is the reference corpus. In the Table 7 below, we show the evaluation results (basic (bas) and cumulative (cum) values) of the Stanford Parser using the measure ParsEval at the parse tree level:

Table 7: Performance degrees of the Stanford parse trees

Length	Precision		Recall		F-measure		Cross-brackets		Time (sec)
	bas	cum	bas	cum	bas	Cum	bas	cum	
[1, 10]	82,24	-	80,96	-	81,59	-	0,84	-	12
[11, 20]	78,80	80,52	76,98	78,97	77,88	79,74	3,79	2,31	25
[21, 30]	72,80	77,95	72,25	76,73	72,53	77,33	6,75	3,79	63
[31, 40]	70,08	75,98	69,75	74,99	69,92	75,48	9,35	5,18	110

[6] http://nlp.cs.nyu.edu/evalb/ (of Sekine, S. and Collins, M. in 2006)

Table 7 shows that the best-marked values of the Stanford parses do not exceed 82,24%. These values deteriorate by about 12% with the sentences whose length is between 30 and 40 words. These values stabilize when the corpus accumulates the most of the sentences. The F-measure score value, that collects the measures recall and precision, reaches 75.48%. Just like the recall and precision values, the values of the cross-brackets measure increase negatively proportionally with the sentence length in the corpus. We recall that this measure calculates the percentage of the syntactic structures in the Stanford parsing result that overlap with those of the reference corpus. Therefore, the lower the value, the better the segmentation in the corpus is according to the reference corpus. The parsing runtime also incrementally increase with the sentence length (9 times the runtime of the sentences of [1, 10] (12/110)). The observation of the ratio between the values of precision and recall for the same division of sentences indicates that these values are very close together for most divisions. Therefore, the Stanford Parser is as relevant as it is accurate.

At the property level, to compare the ATB properties (the reference) to the Stanford properties, we should first enrich the adjusted ATB with the properties of a PG tagged according to the PTB. After that, we enrich the Stanford parse trees with the properties of this PG. We use the measure of [Bensalem 2018] to calculate the similarity degree between properties. It uses the scores of Precision (P) and the Recall(R) (as shown in Table 8).

Table 8: Performance degrees of the properties that describe Stanford parse trees

Length	Const		Uniq		Oblig		Lin		Req		Excl		Σ	
	P	R	P	R	P	R	P	R	P	R	P	R	P	R
[1, 10]	72,6	71,8	64,5	80,3	66,7	67,2	28,8	28,1	0,0	0,0	63,6	63,7	65,8	66,1
[11, 20]	75,3	73,8	64,8	65,1	68,0	66,9	39,6	35,1	10,5	9,5	65,4	65,0	68,0	67,0
[21, 30]	69,9	70,0	64,7	71,1	61,6	62,5	35,0	33,3	33,3	27,9	60,0	62,1	62,5	63,7
[31, 40]	65,4	65,5	60,6	66,1	58,8	59,5	24,0	22,8	18,5	17,2	57,6	58,7	59,2	59,9

From Table 8, we can notice that the precision and recall values for the properties are not always close as for syntactic trees. At times, the precision remarkably exceeds the recall as well as the linearity properties for the division [11, 20] and the requirement properties for the division [21, 30]. Sometimes we find the opposite, that is, the recall takes the first position. This is particularly the case of the uniqueness properties for all divisions except [11, 20] and the exclusion properties for the division [21, 30]. This indicates that the relevance of the Stanford Parser dominates its accuracy. We observed also that the Stanford parse trees include much more unary properties (constituency, uniqueness and obligation). By contrast, we cannot observe any requirement property in the division [1, 10] for example. That is because the ATB uses frequently rules whose syntactic categories include a single lexical category (i.e. NP → NOUN, ADJP → ADJ). The properties on these unique constituents can only be unary and are often satisfied. We cannot have directly these two lexical categories in the same structure, as Stanford Parser does, but through the syntactic categories NP and ADJP. However, the binary properties describe the syntactic categories and not lexical categories in the GP.

5.3.2 Comparison of our analyzer with Stanford Parser

We can compare directly our parses to those of the ATB, since they have the same tagging category and the same transliterated writing. This comparison is also at the two levels: parse tree and property. We use also the evaluation measures of the ParsEval tool at the parse tree level. Table 9 shows the results of this evaluation.

Table 9: Performance degrees of the parse trees of Stanford parser (Stanf) versus our Parser (Ours)

Length	Precision		Recall		F-measure		Cross-brackets		Time (sec)	
	Stanf	Ours	Stanf	Ours	Stanf	Ours	Stanf	Ours	Stanf	Ours
[1, 10]	82,24	86,81	80,96	70,20	81,59	77,62	0,75	2,26	12	7
[11, 20]	78,80	75,52	76,98	67,47	77,87	71,26	3,79	13,11	25	26
[21, 30]	72,80	65,23	72,25	59,45	72,52	62,20	6,75	17,58	63	84
[31, 40]	70,08	61,68	69,75	52,18	69,91	56,53	9,35	20,43	110	193

Table 9 shows that our parser is more accurate than Stanford Parser for the division [1, 10]. However, the recall and the cross-brackets are lower. This indicates that our parser generates almost the full volume of annotations. By contrast, they are not all correct annotations. In addition, for all divisions, the values of the high cross-brackets show that our grouping of lexical categories into segments is not correct, unlike that in the Stanford Parser. The runtime of our parser for the division [1, 10] is lower. It exponentially increases to exceed that of Stanford Parser by about 44%. It is also important to note that for our parser, we devote this time to generate the parse trees and implicitly their properties. However, Stanford Parser does these two tasks separately. The enrichment of its parse trees with properties takes even longer time.

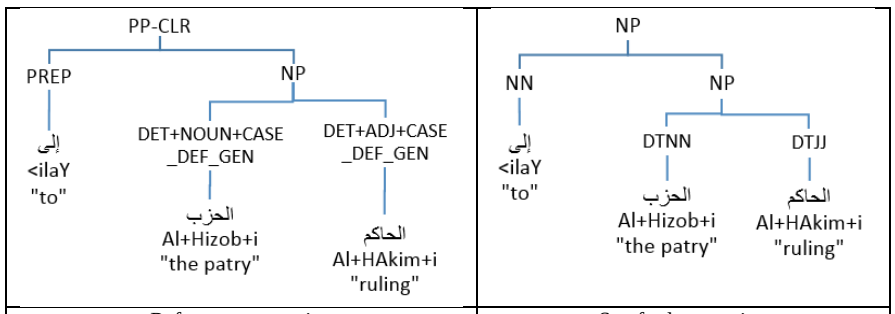
As mentioned in Section 3, the Arabic has linguistic specificities that could make the parsing more ambiguous. For that, we choose to present, in Table 10, the parsing results of more specific Arabic phenomena related to only syntactic ambiguities such as the distinction between the sentence types or the parsing of the NP variants.

Table 10: Performance degrees of Arabic ambiguous phenomena in parse trees of Stanford parser versus our Parser

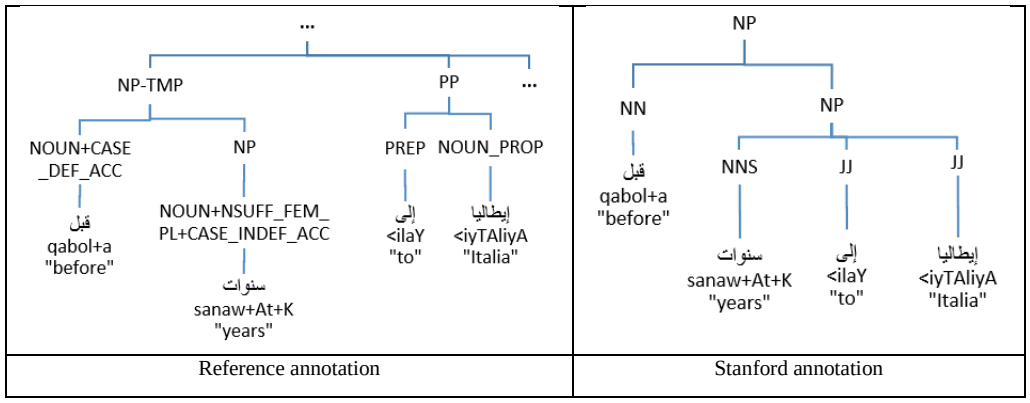
	Stanford Parser				Our Parser			
	P	R	F1	CB	P	R	F1	CB
Verbal sentences	81,86	80,65	81,25	0,73	88,34	74,42	80,78	2,51
Nominal sentences	84,87	83,79	84,32	1,50	84,05	73,59	78,47	3,65
Nominal phrases	74,00	70,44	72,17	0,81	75,23	71,49	73,31	4,08

From the results of Table 10, we observe that we have the best precision (P) for parses of the verbal sentences and the nominal phrases. However, our recall (R) remains lower than Stanford for all sentence types. The nominal phrases are by contrast slightly more pertinent and precise. This indicates that the wrong annotations of our parser are related on other ambiguity types like relatives. This may justify the bad results of cross-brackets (CB). Our encouraging results on the nominal phrases show that our parser deals some related ambiguities better than Stanford parser. This is particularly the case of lexical errors.

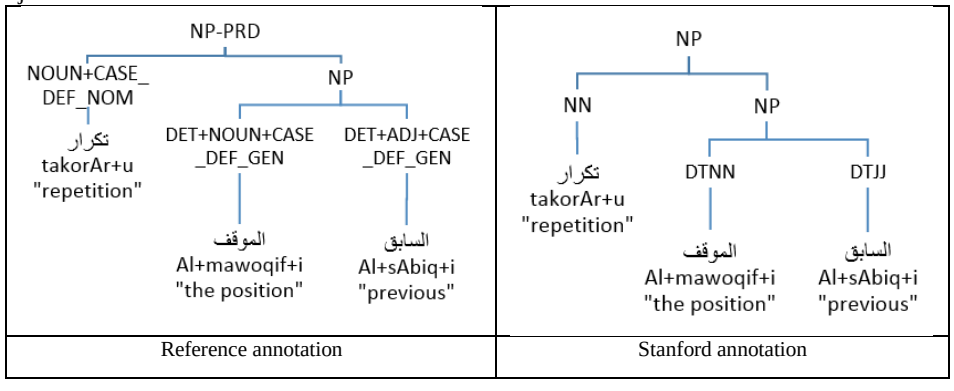
Let's give, as example, the following Prepositional Phrase (PP), composed of three words. Although the first word "إلى" (<ilaY, "to") is a preposition (tagged as PREP), Stanford parser tags it as Noun (NN), which leads to a wrong structure representation (NP) instead of a PP structure. However, our parser generate it as the reference.



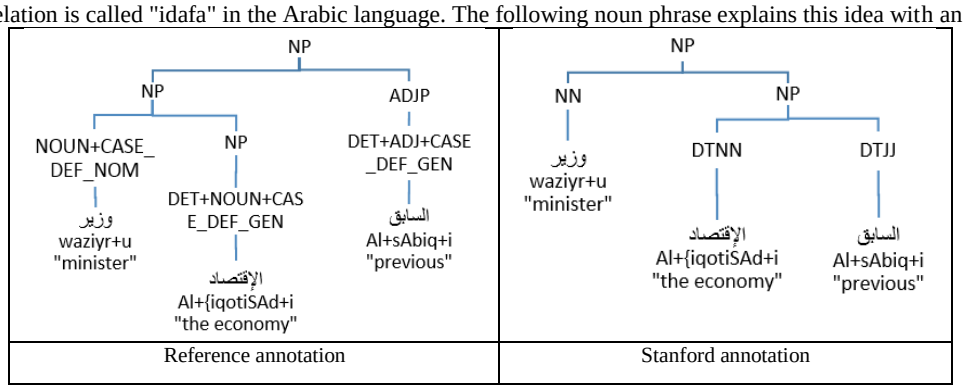
In the following example, not only the word "إلى" (<ilaY, "to") has a wrong POS-tag with Stanford parser, but also the word "إيطاليا" (<iyTAliyA, "Italia"). It generates a wrong annotation and structure.



The following example shows, by contrast, a correct annotation for both Stanford parser and our parser. This annotation relates the second noun in the NP with its following adjective.



However, the same annotation could be wrong with the same POS-tags when the second noun should be related to the first noun according to the context of the text. This relation is called "idafa" in the Arabic language. The following noun phrase explains this idea with an example:



Indeed, both our parser and Stanford parser could not resolve this ambiguity. This is because we need to integrate a higher parsing level: the semantic level. If we observe this form of annotation, we could notice that the first word "وزير" (waziyr+u, "minister") could be replaced by any profession. We could for that resolve this ambiguity by integrating for example a list of professions or other contexts as keywords (as Named Entities) to be verified before the syntactic parsing step. This list could also be defined from the property grammar by the integration of properties between not only the grammatical categories but also the words.

The parsing result of entire sentences may summarize the accuracy degree of our parser as shown in Figure 3, Figure 4 and Figure 5. These figures give also the transliterated format and the English translation of the Arabic words. The parses appear in the left-to-right direction.

<table>
<tr><td>And</td><td>it was completed</td><td>the selection</td><td>slogan</td><td>company</td><td>new</td><td>the week</td><td>last</td></tr>
<tr><td>و</td><td>تم</td><td>اختيار</td><td>شعار</td><td>الشركة</td><td>الجديد</td><td>الأسبوع</td><td>الماضي</td></tr>
<tr><td>wa-</td><td>tam~+a</td><td>{ixotiyAr+u-</td><td>$iEAr+i</td><td>Al+$arik+ap+i</td><td>Al+jadiyd+i</td><td>Al+>usobuwE+a</td><td>Al+mADiy</td></tr>
<tr><td rowspan="6">CONJ</td><td rowspan="5">PV+PVSUFF_SUBJ:3MS</td><td rowspan="4">NOUN+CASE_DEF_NOM</td><td>NOUN+CASE_DEF_GEN</td><td>DET+NOUN+NSUFF_FEM_SG+CASE_DEF_GEN</td><td rowspan="2">DET+ADJ+CASE_DEF_GEN</td><td>DET+NOUN+CASE_DEF_ACC</td><td>DET+ADJ</td></tr>
<tr><td colspan="2">NP</td><td colspan="2" rowspan="2">NP-TMP</td></tr>
<tr><td colspan="3">NP</td></tr>
<tr><td colspan="5">NP</td></tr>
<tr><td colspan="6">NP-SBJ</td></tr>
<tr><td colspan="7">VP</td></tr>
<tr><td colspan="8">S</td></tr>
</table>

Figure 3: Parse tree of the sentence "و تم اختيار شعار الشركة الجديد الأسبوع الماضي" with our parser

The first parse tree indicates that our parser generates correct Nominal phrases (NP) even with nest NP. Our parser considers also the type of the NP such as temporal NP (TMP-NP) and subject NP (SBJ-NP).

<table>
<tr><td>It aims</td><td>to</td><td>oppose</td><td>regime</td><td>governance</td><td>the existing</td><td>in</td><td>the country</td></tr>
<tr><td>يهدف</td><td>إلى</td><td>مناهضة</td><td>نظام</td><td>الحكم</td><td>القائم</td><td>في</td><td>البلاد</td></tr>
<tr><td>ya+hodif+u</td><td><ilaY</td><td>munAhaD+ap+i</td><td>niZAm+i</td><td>Al+Hukom+i</td><td>Al+qA}im+i</td><td>fiy-</td><td>Al+bilAd+i</td></tr>
<tr><td>IV3MS+IV+IVSUFF_MOOD:I</td><td rowspan="5">PREP</td><td rowspan="3">NOUN+NSUFF_FEM_SG+CASE_DEF_GEN</td><td rowspan="2">NOUN+CASE_DEF_GEN</td><td>DET+NOUN+CASE_DEF_GEN</td><td>DET+ADJ+CASE_DEF_GEN</td><td>PREP</td><td>DET+NOUN+CASE_DEF_GEN</td></tr>
<tr><td colspan="3">NP</td><td colspan="2">PP</td></tr>
<tr><td></td><td colspan="4">NP</td></tr>
<tr><td></td><td colspan="5">NP</td></tr>
<tr><td></td><td colspan="6">PP-CLR</td></tr>
<tr><td colspan="8">VP</td></tr>
<tr><td colspan="8">S</td></tr>
</table>

Figure 4: Parse tree of the sentence "يهدف إلى مناهضة نظام الحكم القائم في البلاد" with our parser

Figure 4 shows correct parses of the Prepositional Phrases (PP) with the precision of the type CLR (CLaused Related to phrasal verbs such as prepositional ditransitives). However, empty pronouns are not considered by our parser. So, it does not annotate the empty subject (as Stanford parser).

In the same example, the annotation of the last NP is not correct with our parser, because it considers the structure "في البلاد" independent of the noun "القائم". Stanford parser make the correct analysis.

<table>
<tr><td>يتم</td><td>إستخدام</td><td>ـه</td><td>في</td><td>إصلاح</td><td>الاختلالات</td><td>الهيكلية</td><td>في</td><td>المصرف</td></tr>
<tr><td>ya+tim~+u</td><td>{isotixodAm+u-</td><td>-hu</td><td>fiy-</td><td><iSolAH+i</td><td>Al+{ixotilAl+At+i</td><td>Al+hayokaliy~+ap+i</td><td>fiy-</td><td>Al+maSorif+i</td></tr>
<tr><td rowspan="6">IV3MS+IV+IVSUFF_MOOD:I</td><td rowspan="2">NOUN+CASE_DEF_NOM</td><td rowspan="2">POSS_PRON_3MS</td><td rowspan="5">PREP</td><td rowspan="3">NOUN+CASE_DEF_GEN</td><td>DET+NOUN+NSUFF_FEM_PL+CASE_DEF_GEN</td><td>DET+ADJ+NSUFF_FEM_SG+CASE_DEF_GEN</td><td rowspan="2">PREP</td><td>DET+NOUN+CASE_DEF_GEN</td></tr>
<tr><td colspan="2">NP</td><td>PP</td></tr>
<tr><td colspan="3">NP</td></tr>
<tr><td colspan="2">NP</td><td colspan="4">PP</td></tr>
<tr><td colspan="6">NP-SBJ</td></tr>
<tr><td colspan="8">VP</td></tr>
<tr><td colspan="8">S</td></tr>
</table>

Figure 5: Parse tree of the sentence " يتم إستخدامه في إصلاح الاختلالات الهيكلية في المصرف" with our parser

Figure 5 shows another example in which our parser generates a wrong construction. The subject here concerns only the NP "إستخدامه", but not the following PP. Stanford parser generates a parse tree as the reference. We may justify the mentioned wrong annotations by many reasons, namely:

- Our parser do not consider the empty pronouns which could be a subject
- The CNF of our learning model may affect the choice of the better production rule.
- The lack of constraints that consider more Arabic specificities.

To evaluate the generated properties that describe our parse trees, we enrich the ATB reference sentences with the properties of the PG at the maximum granularity [Bensalem 2014a]. We use the enrichment technique described in [Bensalem 2016]. The following Table 11 illustrates the evaluation results of our parser compared with Stanford Parser at the property level.

Table 11: Performance degrees of the properties that describe the parse trees of Stanford Parser (Stanf) versus our parser (Ours)

Type	Const				Uniq				Oblig			
	Precision		Recall		Precision		Recall		Precision		Recall	
Division	Stanf	Ours	Stanf	Ours	Stanf	Our	Stanf	Ours	Stanf	Our	Stanf	Ours
[1, 10]	71,34	79,95	68,67	68,41	64,47	76,36	80,33	71,48	66,67	80,05	67,15	79,84
[11, 20]	75,29	76,47	73,82	65,59	64,8	73,81	65,13	72,28	67,96	78,32	66,85	76,01
[21, 30]	69,86	74,55	70,01	64,24	64,69	70,69	71,13	68,47	61,63	75,13	62,45	75,67
[31, 40]	65,39	72,82	65,53	61,38	60,64	69,64	66,13	65,11	58,78	74,07	59,51	73,09
Type	Lin				Req				Excl			
	Precision		Recall		Precision		Recall		Precision		Recall	
Division	Stanf	Ours	Stanf	Ours	Stanf	Ours	Stanf	Ours	Stanf	Ours	Stanf	Ours
[1, 10]	28,83	49,54	28,07	46,14	0,00	44,73	0,00	42,47	63,64	77,42	63,73	75,44
[11, 20]	39,64	46,98	35,05	42,39	10,53	38,69	9,52	37,22	65,43	72,09	65,01	69,46
[21, 30]	34,98	41,75	33,26	35,07	33,33	32,81	27,91	29,48	60,00	68,71	62,12	61,29
[31, 40]	24,01	35,32	22,78	26,49	18,52	26,94	17,24	24,36	57,59	61,82	58,65	57,56

According to Table 11, we can notice that unlike its parse tree evaluation, our parser provides for most property types higher performances than Stanford Parser. Even for the requirement properties, our parser expresses it sufficiently for the division [1, 10] although they correctly describe a lot of syntactic categories. That is true for all binary properties. We can explain these findings by two reasons: Our parser build the trees using rules whose probabilities are learned on the property frequencies, and they have the same form of the reference corpus rules.

6 OPEN PROBLEMS

6.1 Generated properties

There are other forms of relations that could be observed for Arabic sentences such as those of [Habash 2009]. We started with the automatically induced relations. In the perspectives, we will integrate multi-level relations even those that require rules to apply.

6.2 Combination with BERT

The pre-trained model BERT provides good results. The combination of it with our parser could be a good alternative. Therefore, we present the common and difference points with it. BERT parser, as pre-trained model, runs an encoder to read the entire input sequence of words bidirectionally. It learns contextual relations between all words using the technique of masking. To predict a masked word of a sentence, it runs a CNN training of these bidirectional contextual relations. The input words should be tokenized and expressed into sentence embeddings and positional embeddings. In our parser, we represent the grammatical constructions also into token sequences with their positions and the first and the last tokens. We use this information to generate the heads (or obligation properties) of the constructions by running the rule system of Habash ([Habash 2009]). However, we read words only in a left-to-right direction. We learn from the constructions also other syntactic relations (or properties). The advantage of the property grammar formalism that we followed is that we have simple, direct and local representation independently of the type, the context or the position of the information. This formalism is flexible and robust, so that even if the information is incomplete, it could generate possible relations. In addition, this formalism expresses

information directly on categories and do not require building a local structure for the syntactic information before using the described constraints. Other constraint-based approaches need such structure (i.e. local trees for HPSG and dependency relation for CDG [Ben Ismail 2019, Dekhtyar 2015]). The second phase in the parser approach is the application of a parsing algorithm to select the best combination of these constructions.

6.3 Arabic Ambiguities

The specificity of the Arabic language could make its processing very difficult, and generates different morphological and syntactic ambiguities. The latter are especially caused by the agglutination, the absence of vocalization and the relatively free word order in the Arabic sentences. The agglutination is very frequent in Arabic and could not be easily detected. For example, the character "فـ" (pronounced as "f") is original in the word "فِرَقٌ" (firaqN, “teams”) but it is an attached particle in the prefix of the word "فَرَقَ" (fa+raq~a, "then he becomes tender"). Without vocalization, the morphological analyzer consider these two words as the same word. This ambiguity remains also with the possessive and objective pronouns that are attached as suffix to the word. For example, the word "كتبه" without vocalization could have two possible annotations:

- The annotation of "كُتُبَهُ", which is a nominal phrase that includes a noun followed by its possessive pronoun (kutuba+hu, "his books") or,
- The annotation of "كَتَبَهُ", which is a verbal sentence that includes a verb followed by an inferred pronoun as subject and an object pronoun (kataba+hu, "he writes it").

In addition, unlike English, there are in the Arabic two types of sentences with different annotations: the nominal sentence, which includes a topic and a predicate, and the verbal sentence, which includes a verb, a subject and generally an object. The nominal phrases have also different forms, they can include or not coordination. Their units could have or not the same case, gender or number. The absence of vocalization often make ambiguity about the functions of words in the sentence because this affects the case of the word. Thus, the subject is nominative, the object is accusative, and a word that follows a preposition is genitive. There are specific particles that could also affect the case of the words. They could be ambiguous if they are not vocalized. For example, the expression "لم تذهب" could have two possible annotations:

- The annotation of "لَمْ تَذْهَبْ" (lam ta*°hab°, "you did not go"), in which the negation particle "لم" (lam, "not") makes the present verb "تذهب" in the jussive case.
- The annotation of "لِمَ تَذْهَبُ" (lima ta*°habu, "why are you going") that includes the interrogation noun "لم" (lima, "why") and the present verb "تذهب" in the indicative case.

The Arabic have also inferred pronoun (ضمير مستتر), it replaces the subject that is not observed in the sentence.

As mentioned in [Ababou et al., 2017], the existing parser results are still insufficient in terms of syntactic information because they not consider enough the syntactic functions specific to the Arabic language. To do that, according to the previously indicated work, the analysis should combine a morphological analysis, a unification grammar formalism and a traditional grammar. This paper will be a starting point to study in-depth all the specific syntactic functions of the Arabic.

6.4 Weakness compared to Stanford parser

There are three reasons making Stanford parser more performant to ours. Each one could potentially be solved to have better results:

- Our parser do not consider the empty pronouns which could be a subject: Empty pronoun will be predicted based on adjacent word agreement.
- The CNF of our learning model may affect the choice of the better production rule : We may apply a non CNF-based parsing algorithm (such as Earley algorithm).
- The lack of constraints that consider more Arabic specificities : There are another property types such as idafa and adjacency that could well take into account the Arabic specifies.

7 CONCLUSION AND PERSPECTIVES

We described in the present paper a probabilistic property parser, that provides new information in addition to the hierarchical relations given by the parse trees. The comparison of the results to the Stanford Parser is on two levels: parse trees and local properties. Our parser uses a very informative learning model that exploits the production rules, implicit information namely the lexical category sequences, the heads of each syntactic category and

finally the syntactic properties. We obtain this information by applying an in-depth analysis and the lexicalization of the syntactic categories. This lexicalization informs us about the content of the ATB syntactic structures. We collected information like the most frequent heads in the ATB, which could guide the parsing process.

As perspectives, since our property parsing technique is automatic, we can reuse it for treebanks of different languages. In addition, in order to offer a very precise representation of the syntactic information, we can always enrich or modify the set of relations presented in our grammars. Indeed, the large and abnormal number of the exclusion properties can be reduced by proposing new interpretations that address a set of factors such as the position in the structure, the category of the structure, its occurrences, and the symmetry between the units of the relation. It is also possible to add interpretations to new types of syntactic properties such as the agreement and the dependencies. The properties can cover in addition other analysis levels (e.g. semantic and morphological). Profiting from the various formats of the treebank can help to extract information like the English translation of an Arabic word. This allows enlarging the axes of semantic and morphological analysis of the Arabic words. We also plan to take advantage of the syntactic properties to build classification-based learning models. These properties may be the classification criteria of the syntactic structures. They need a digital representation in such case. We can go further by transforming our property parser into a hybrid version. This version combines a set of rules and keywords lists, which are prepared manually by experts with the learning model rules automatically induced from the ATB. Using this, we wish to strengthen the processing of the particular linguistic phenomena in the future parser.

REFERENCES

[1] P. Blache. 2016. Representing Syntax by Means of Properties: a Formal Framework for Descriptive Approaches. Journal of Language Modelling, 4 , 2, 183-224. DOI: https://doi.org/10.15398/jlm.v4i2.129.

[2] P. Blache, and S. Rauzy. 2006. Mécanismes de contrôle pour l'analyse en Grammaires de Propriétés. P. Mertens, C. Fairon, A. Dister et P. Watrin, 415-424.

[3] M. Maamouri, A. Bies, and S. Kulick. 2008. Enhancing the Arabic treebank: a collaborative effort toward new annotation guidelines. In Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08), European Language Resources Association (ELRA), Marrakech, Morocco.

[4] S. Kulick, A. Bies, and M. Maamouri. 2010. Consistent and flexible integration of morphological annotation in the Arabic treebank. In Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10) European Language Resources Association (ELRA), Valletta, Malta.

[5] N. Khoufi, C. Aloulou, and L. Belguith. 2014. Supervised learning model for parsing arabic language. ArXiv: abs/1410.8783.

[6] N. Khoufi, C. Aloulou, and L. Belguith. 2016. Toward hybrid method for parsing modern standard Arabic. In Proceedings of the 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), Shanghai, China, 451-456. DOI: 10.1109/SNPD.2016.7515939.

[7] A. B. Soliman, K. Eissa, and S. El-Beltagy. 2017. Aravec: A set of arabic word embedding models for use in arabic nlp, Procedia Computer Science,. arabic Computational Linguistics, 256-265. DOI: 10.1016/j.procs.2017.10.117.

[8] D. Chen, and C. Manning. 2014. A fast and accurate dependency parser using neural networks. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Doha, Qatar, 740-750. DOI: 10.3115/v1/D14-1082.

[9] J. Nivre, M.-C. de Marnee, F. Ginter, Y. Goldberg, J. Hajic, C. D. Manning, R. McDonald, S. Petrov, S. Pyysalo, N. Silveira, R. Tsarfaty, and D. Zeman. 2016. Universal Dependencies v1: A multilingual treebank collection. In: Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16), European Language Resources Association (ELRA), Portoroz, Slovenia, 1659-1666.

[10] T. Dozat, P. Qi, and C. D. Manning.2017. Stanford's graph-based neural dependency parser. In: Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, Association for Computational Linguistics, Vancouver, Canada, 20-30. DOI: 10.18653/v1/K17-3002.

[11] M. Fouad, A. Mahany, N. Aljohani, R. Abbasi, and S.-U. Hassan. 2020. Arwordvec: efficient word embedding models for arabic tweets, Soft Computing 24. DOI: 10.1007/s00500-019-04153-6.

[12] W. Che, Y. Liu, Y. Wang, B. Zheng, and T. Liu. 2018. Towards better UD parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. In: Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, Association for Computational Linguistics, Brussels, Belgium, 55-64. DOI: 10.18653/v1/K18-2005.

[13] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer.2018. Deep contextualized word representations. ArXiv:1802.05365.

[14] D. Zeman, J. Hajic, M. Popel, M. Potthast, M. Straka, F. Ginter, J. Nivre, and S. Petrov. 2018. CoNLL 2018 shared task: Multilingual parsing from raw text to Universal Dependencies in: Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies, Association for Computational Linguistics, Brussels, Belgium, 1-21. DOI: 10.18653/v1/K18-2001.

[15] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Association for Computational Linguistics, Minneapolis, Minnesota, 4171-4186. DOI: 10.18653/v1/N19-1423.

[16] N. Kitaev, S. Cao, and D. Klein. 2019. Multilingual constituency parsing with self-attention and pre-training. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Florence, Italy, 3499-3505. DOI: 10.18653/v1/P19-1340.

[17] N. Kitaev, and D. Klein. 2018. Constituency parsing with a self-attentive encoder. ArXiv:1805.01052.

[18] A. Safaya, M. Abdullatif, and D. Yuret. 2020. KUISAIL at SemEval-2020 task 12: BERT-CNN for offensive speech identification in social media. In: Proceedings of the Fourteenth Workshop on Semantic Evaluation, International Committee for Computational Linguistics, Barcelona (online), 2054-2059.

[19] W. Antoun, F. Baly, and H. Hajj. 2021. Arabert: Transformer-based model for arabic language understanding. ArXiv:2003.00104.

[20] J. Howard, and S. Ruder. 2018. Universal language model fine-tuning for text classification. ArXiv:1801.06146.

[21] J. M. Eisenschlos, S. Ruder, P. Czapla, M. Kardas, S. Gugger, and J. Howard. 2019. Multit: Efficient multi-lingual language model fine-tuning. ArXiv:1909.04761.

[22] M. Artetxe, and H. Schwenk. 2019. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond, Transactions of the Association for Computational Linguistics 7, 597-610. DOI: 10.1162/tacl a 00288.

[23] D. Duchier, T.-B.-H. Dao, Y. Parmentier, and W. Lesaint. 2010. Une modélisation en CSP des grammaires de propriétés. In: JFPC 2010 – Sixièmes Journées Francophones de Programmation par Contraintes, Caen, France, 123-132.

[24] D. Duchier, T.-B.-H. Dao, and Y. Parmentier. 2012. Analyse Syntaxique par Contraintes pour les Grammaires de Propriétés à traits. In: Huitièmes Journées Francophones de Programmation par Contraintes (JFPC 2012), Toulouse, France, 101-106.

[25] P. Blache, and S. Rauzy. 2014. A chinese constraint grammar extracted from the chinese treebank. In: Proceedings of Second Asia Pacific Corpus Linguistics Conference (APCLC 2014), Hong Kong.

[26] R. B. Bensalem, M. Elkarwi, K. Haddar, and P. Blache. 2014. Building an Arabic linguistic resource from a treebank: the Case of Property Grammar. In: 17th International Conference on Text, Speech and Dialogue (TSD 2014), Vol. 8655, Springer, Brno, Czech Republic, 240-246.

[27] P. Blache, and S. Rauzy. 2012. Hybridization and Treebank Enrichment with Constraint-Based Representations. In: LREC-2012, Istanbul, Turkey, 6-13.

[28] P. Blache, and S. Rauzy. 2012. Enrichissement du FTB : un treebank hybride constituants/propriétés. In: TALN-2012, Vol. 2, Grenoble, France, 307-320.

[29] R. Bensalem, K. Haddar, and P. Blache. 2016. A property grammar-based method to enrich the Arabic treebank ATB. In: Knowledge Discovery, Knowledge Engineering and Knowledge Management, Springer, 302-323. DOI: 10.1007/978-3-319-52758-1 17.

[30] R. B. Bensalem, K. Haddar, and P. Blache. 2015. A formal modeling method to enrich the arabic treebank atb with syntactic properties. In: Proceedings of the International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management, IC3K 2015, SCITEPRESS - Science and Technology Publications, Lda, Lisbon, Portugal, 108-117. DOI: 10.5220/0005617001080117.

[31] R. B. Bensalem, N. Kadri, K. Haddar, and P. Blache. 2018. Evaluation and enrichment of stanford parser using an arabic property grammar. In: Computational Linguistics and Intelligent Text Processing (CICLing 2017), Springer International Publishing, Hungary, 170-182. DOI: 10.1007/978-3-319-77113-7 14.

[32] P. Blache, S. Rauzy, and G. Montcheuil. 2016. MarsaGram: an excursion in the forests of parsing trees. In: Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16), European Language Resources Association (ELRA), Portoroz, Slovenia, 2336-2342.

[33] I. Adebara. 2019. Womb grammars: A constraint solving model for learning the grammar of yoruba. In: Proceedings of European Language Resources Association (ELRA), Paris, France, 169-172.

[34] T. Kasami. 1965. An efficient recognition and syntax analysis algorithm for context-free languages (AFCRL-65-758).

[35] D. H. Younger. 1967. Recognition and parsing of context-free languages in time n3, Information and Control 10 ,2,189-208. DOI: 10.1016/S0019 -9958(67)80007-X.

[36] N. Habash, R. Faraj, and R. Roth. 2009. Syntactic annotation in the columbia arabic treebank. In: Proceedings of the International Conference on Arabic Language Resources and Tools (MEDAR), Cairo, Egypt.

[37] J. Earley. 1970. An efficient context-free parsing algorithm, Commun. ACM 13,2,94-102. DOI: doi:10.1145/362007.362035.

[38] S. Ben Ismail, S. Boukédi, and K. Haddar. 2019. HPSG Grammar Supporting Arabic Preference Nouns and Its TDL Specification. In: K. Smaïli, (eds) Arabic Language Processing: From Theory to Practice. ICALP2019. Communications in Computer and Information Science, Vol. 1108. Springer, Cham. DOI: https://doi.org/10.1007/978-3-030-32959-4_16

[39] M. Dekhtyar, A. Dikovsky, and B. Karlov. 2015. Categorial dependency grammars, Theoretical Computer Science, Volume 579,33-63, ISSN 0304-3975 DOI: https://doi.org/10.1016/j.tcs.2015.01.043.

[40] D. Abdelrazaq, S. Abu-Soud, A. Awajan, and Arafat. 2018., A Machine Learning System for Distinguishing Nominal and Verbal Arabic Sentences. The International Arab Journal of Information Technology (IAJIT), 15,3,567-584.

[41] R. Maalej, N. Khoufi, and C. Aloulou. 2021. Parsing Arabic using deep learning technology. In Proceeings of the Tunisian-Algerian Joint Conference on Applied Computing (TACC), Tabarka, Tunisia, 74-80.

[42] S. Al-Ghamdi, H. Al-Khalifa, and A. Al-Salman. 2021. A Dependency Treebank for Classical Arabic Poetry. In Proceedings of the Sixth International Conference on Dependency Linguistics (Depling, SyntaxFest), 1–9, Sofia, Bulgaria, Association for Computational Linguistics.

[43] N. Habash, M. AbuOdeh, D. Taji, R. Faraj, J. El Gizuli, and O. Kallas. 2022. Camel Treebank: An Open Multi-genre Arabic Dependency Treebank. In Proceedings of the Thirteenth Language Resources and Evaluation Conference, 2672–2681, Marseille, France. European Language Resources Association.

[44] S. Al-Ghamdi, H. Al-Khalifa, and A. Al-Salman. 2023. Fine-Tuning BERT-Based Pre-Trained Models for Arabic Dependency Parsing, Applied Sciences, 13, 7, 4225 DOI: https://doi.org/13.4225. 10.3390/app13074225.

[45] D. Aqel, Darah, S. AlZu'bi, and S. Hamadah. 2019. Comparative Study for Recent Technologies in Arabic Language Parsing. In Proceedings of the sixth International Conference on Software Defined Systems (SDS), 209-212. DOI: https://doi.org/10.1109/SDS.2019.8768587.

[46] Y. M. Saber, H. Abdel-Galil, and M. A. Belal. 2022. Arabic ontology extraction model from unstructured text. Journal of King Saud University - Computer and Information Sciences, 34,8, Part B,6066-6076, ISSN 1319-1578. DOI: https://doi.org/10.1016/j.jksuci.2022.02.007.

[47] A. Morad, M. Nagi, and S. Alansary. 2021. Deep Learning–Based Constituency Parsing for Arabic Language. In: Arabnia, H.R. Ferens, K. de la Fuente, D. Kozerenko, E.B. Olivas Varela, J.A. Tinetti, F.G. (eds) Advances in Artificial Intelligence and Applied Cognitive Computing, Transactions on Computational Science and Computational Intelligence, Springer, Cham, 45–58. DOI: https://doi.org/10.1007/978-3-030-70296-0_4.

[48] N. Ababou, A. Mazroui, and R. Belehbib. 2017. Parsing Arabic Nominal Sentences Using Context Free Grammar and Fundamental Rules of Classical Grammar. International Journal of Intelligent Systems and Applications, 9,8, 11 pages.

A APPENDIX

Arabic BERT-based parsers

- The last Berkeley parser [Kitaev 2019], which provides a joint multilingual pre-training and fine-tuning method that shares most of the parameters between ten languages, including Arabic. This method [Kitaev 2018] combines unsupervised pre-trained models of BERT with a neural attention architecture, which increases its accuracy. This attention architecture makes explicit how the information is transferred between different locations in the sentence, based on their positions, and their contents.

- Arabic BERT [Safaya 2020], which feeds the outputted contextualized embeddings of the last four hidden layers of the BERT model into 160 convolutional filters for several layers of the CNN. The output of the CNN is then passed to a dense layer to obtain the predictions. Arabic BERT was trained on 8000 offensive and non-offensive tweets and tested on 2000 ones to generate 89.7% as F1-score.
- AraBERT [Antoun 2021], which has 110 million parameters mirroring the base BERT model. As well as BERT, the pre-training setup for AraBERT includes the two unsupervised tasks of Masked Language Model (MLM) and Next Sentense Prediction (NSP). The pre-training data was about 700 million sentences, and the tokenization was resulted in a vocabulary size of 64K words, derived from sources such as the 1.5 billion words Arabic Corpus and OSIAN: the Open Source International Arabic News Corpus that consists of 3.5 million articles. AraBERT is applied to three NLU downstream tasks: Sentiment Analysis (with 96.2%), Named Entity Recognition (with 84.2%) and Question Answering (with 93.0%).