Universal Dependencies for Arabic

Dima Taji, Nizar Habash and Daniel Zeman†
Computational Approaches to Modeling Language (CAMeL) Lab,
New York University Abu Dhabi
†Faculty of Mathematics and Physics, Charles University
{dima.taji,nizar.habash}@nyu.edu, zeman@ufal.mff.cuni.cz

Abstract

We describe the process of creating NUDAR, a Universal Dependency treebank for Arabic. We present the conversion from the Penn Arabic Treebank to the Universal Dependency syntactic representation through an intermediate dependency representation. We discuss the challenges faced in the conversion of the trees, the decisions we made to solve them, and the validation of our conversion. We also present initial parsing results on NUDAR.

1 Introduction

Parsers have been used in many Natural Language Processing (NLP) applications, such as automatic summarization, question answering, and machine translation. This motivates the creation of treebanks on which these parsers can be trained. Treebanks have two main different syntactic representations. On one hand, there are phrase structure (constituency) treebanks such as the Penn Treebank (Marcus et al., 1993), and its sister treebanks such as the Penn Arabic Treebank (PATB) (Maamouri et al., 2004) and the Penn Chinese Treebank (Xue et al., 2005). On the other hand, there are dependency treebanks, such as Columbia Arabic Treebank (CATiB) (Habash and Roth, 2009), and the Prague Dependency Treebank (PDT) (Hajič et al., 2001). Other treebanks that followed the style of PDT are the Slovene (Džeroski et al., 2006) and the Croatian (Berović et al., 2012) treebanks, as well as the Prague Arabic Dependency Treebank (PADT) (Smrž et al., 2002; Hajič et al., 2004; Smrž et al., 2008).

Having these different syntactic representations makes it difficult to compare treebanks, and parsing results (Nilsson et al., 2007). This motivated the creation of the Universal Dependency (UD) syntactic representation, that aims to create cross-linguistically consistent annotation guidelines that facilitate the creation of treebanks that are built with the same label sets and structural basis (Nivre, 2014; Pyysalo et al., 2015). In this paper, we present the *New York University Abu Dhabi Universal Dependency Arabic Treebank*, a UD treebank for Arabic, which we dub NUDAR.[1]

2 Related Work

In this section we present the Universal Dependency syntactic representation, as well as some of the most prominent previous efforts on Modern Standard Arabic (MSA) treebanks.

2.1 Universal Dependencies

UD is an open community effort. It builds on the existing treebank structure of the Stanford dependencies (De Marneffe et al., 2006; De Marneffe and Manning, 2008; De Marneffe et al., 2014), as well as the universal Google dependency scheme (McDonald et al., 2013). In addition, it makes use of the Google Universal Parts-of-Speech (POS) Tagset (Petrov et al., 2011), and the morphosyntactic tag set of the *interset interlingua* (Zeman, 2008).

The aim of UD is to facilitate the creation of treebanks in different languages that are consistent in their syntactic representation, while still allowing the extension of the relations to accommodate for language-specific constructs. The target of UD is to facilitate the development of multilingual learning systems, and multilingual NLP, as well as allow for comparative linguistic studies and evaluation (Nivre et al., 2016).

In its last release of version 1.4, the UD treebank collection contained 64 different treebanks,

[1] The noun Nudar نُضَار *nuDAr* is Arabic for ‘pure gold’.

Proceedings of The Third Arabic Natural Language Processing Workshop (WANLP), pages 166–176,
Valencia, Spain, April 3, 2017. ©2017 Association for Computational Linguistics

with over 10 other treebanks scheduled for release in the upcoming version 2.0. The treebanks are in 47 languages, including Swedish (Nivre, 2014), Danish (Johannsen et al., 2015), Finnish (Pyysalo et al., 2015), Estonian (Muischnek et al., 2014), Norwegian (Øvrelid and Hohle, 2016), Croatian (Agić and Ljubešić, 2015), Persian (Seraji et al., 2016), Bulgarian (Osenova and Simov, 2015), Catalan and Spanish (Alonso and Zeman, 2016), as well as the Prague Arabic Universal Dependency Treebank (PAUDT), among others.

2.2 Arabic Treebanks

A number of treebanks exist for MSA. These treebanks vary in terms of their syntactic representation (constituency vs. dependency), richness of annotation, and source of data. We discuss next four treebanks that are relevant to this paper.

PATB: The Penn Arabic Treebank (Maamouri et al., 2004; Maamouri et al., 2009) is a Linguistic Data Consortium (LDC) project, for which there are currently 12 parts for MSA. PATB consists of constituency trees, the sources of which are newswire articles from a variety of news sources.

CATiB: Columbia Arabic Treebank (Habash and Roth, 2009) is a dependency treebank effort that allows for faster annotation and uses "intuitive dependency structure representation and relational labels inspired by traditional Arabic grammar" (Habash, 2010). The basic CATiB treebank uses six POS tags, and eight relational labels. It contains 273K tokens that have been annotated directly in the CATiB representation, as well as the entire PATB parts 1, 2, and 3 that were automatically converted into CATiB representation.

PADT and PAUDT: The Prague Arabic Dependency Tree 1.0 (Smrž et al., 2008) was published in the LDC in 2004 (Hajič et al., 2004) and consisted of about 114K tokens. The data in PADT comes from part of the PATB parts 1 and 2, and the Arabic Gigaword (Graff et al., 2006). Variants of that dataset were released for the CoNLL 2006 (60K tokens) and CoNLL 2007 (116K tokens, improved morphology) shared tasks. An extended dataset (282K tokens) was incorporated in the HamleDT collection, where 30 treebanks were first harmonized in the Prague annotation style, later in Stanford dependencies (Rosa et al., 2014). Finally, this dataset was converted to Universal Dependencies and it has been part of UD releases since UD 1.2, labeled simply "UD Arabic".[2]

The annotation guidelines of PADT 1.0 were derived from the Prague Dependency Treebank (Czech), with some necessary adjustments to account for the differences between Arabic and Czech. The original morphological and syntactic disambiguation was done manually but the subsequent conversion steps were automatic.

Word forms in PADT are fully diacritized; in PAUDT we preserve the diacritized as a useful extra attribute, but the main form is undiacritized, to provide more natural training material for parsers. Morphological tags were converted to the UD tags and features, dependency relation types were translated to the UD label set. Occasionally the translation of labels relied on other sources such as part of speech or even lemma. For example, the PADT relation `AuxM` *(modifying expression)* is used for prepositions (which are attached as `case` or `mark` in PAUDT), for negative particles *lA, lam, lan*[3] (which ended up as `neg` in UD v1 and as `advmod` in UD v2), for future particles *sa, sawfa* (which are `aux` dependents in PAUDT) and also for the negative copula *laysa* (`cop` in PAUDT).

Unlike UD, the Prague treebanks do not distinguish whether the dependent is a nominal or a clause (`nsubj` vs. `csubj`, `obj` vs. `ccomp` etc.) Heuristics have to be used here. At present, only phrases headed by verbs are considered clausal; clauses with non-verbal predicates without a copula are attached as if they were bare nominals. On the other hand, when a copula is involved, we reattach it as a dependent of the non-verbal predicate (in PADT, if the copula is present, it heads the clause). Similarly, prepositions head prepositional phrases in the Prague style but they are attached as modifiers of their nouns in PAUDT.

Finally, coordination in the Prague style is al-

[2] In this paper we use UD to refer to the general shared concept of Universal Dependency representation. For language specific decision and treebanks we will use the name of the treebanks, i.e. PAUDT or NUDAR.

[3] All Arabic transliterations are provided in the Habash-Soudi-Buckwalter transliteration scheme (Habash et al., 2007b). This scheme extends Buckwalter's transliteration scheme (Buckwalter, 2002) to increase its readability while maintaining the 1-to-1 correspondence with Arabic orthography as represented in standard encodings of Arabic, i.e., Unicode, CP-1256, etc. The following are the only differences from Buckwalter's scheme (which is indicated in parentheses): Ā آ (|), Â أ (>), ŵ ؤ (&), Ǎ إ (<), ŷ ئ (}), ħ ة (p), θ ث (v), ð ذ (*), š ش ($), Ď ظ (Z), ς ع (E), γ غ (g), ý ى (Y), ã ـً (F), ũ ـٌ (N), ĩ ـٍ (K), á ـٰ (').

ways headed either by a conjunction, or, if no conjunction is present, by a punctuation symbol. All conjuncts are at the same tree level. In PAUDT these structures are transformed so that the first conjunct is the head and all subsequent conjuncts are attached to it.

Why Another Arabic Universal Dependency Treebank? PAUDT is based on PADT, which is a small treebank, compared to the existing PATB treebank. Our aim is to make use of the automatic conversion of PATB, parts 1, 2, and 3, into a richer version of CATiB, and use it to create NUDAR. This would allow us in the future to convert the remaining parts of PATB, both in MSA and dialectal Arabic (such as Egyptian (Maamouri et al., 2012)), as well as extend the existing CATiB treebank that has no parallel in PATB's constituency representation.

3 NUDAR: NYUAD Universal Dependency Arabic Treebank

In this section we describe the creation of NUDAR, starting with the PATB. The conversion strategy we adopt is to transform the constituency PATB trees into the rich CATiB++ dependency representation (Section 3.2). We then apply morphological and syntactic transformations on these trees – Section 3.3, and Section 3.4 respectively.

3.1 A Note on Tokenization and Datasets

The datasets that are currently included in NUDAR are the PATB part 1, v4.1 (Maamouri et al., 2010a), part 2, v3.1 (Maamouri et al., 2011), and part 3, v3.2 (Maamouri et al., 2010b). The tokenization followed in NUDAR is the same tokenization scheme followed in PATB, which tokenizes all the clitics, with the exception of the definite article +ال *Al*+ 'the' (Pasha et al., 2014). The treebank contains 19K sentences, containing 738K tokens. For our parsing experiment, we followed the guidelines detailed by Diab et al. (2013), to split the treebanks into TRAIN, DEV, and TEST. The details of the sizes of the different datasets are shown in Table 1.

3.2 From Constituency to Dependency

Our conversion pipeline starts from PATB, converting it to a richer version of the Columbia Arabic Treebank which we refer to as CATiB++. We use the Columbia Arabic Conversion Tool v0.7 (Habash and Roth, 2009),[4] that converts PATB trees to the CATiB representation, with the addition of the semantic dashtags and the PATB complete morphological tags (BW) (Buckwalter, 2004). We supplement the trees with additional feature-value pairs representation in the style used in the MADAMIRA morphological analyzer and disambiguator (Pasha et al., 2014).

We chose to convert the treebanks through this methodology to allow for the conversion of the existing CATiB treebank that has no parallel in PATB's constituency representation. In the future, we envision enriching the CATiB treebank with the morphosyntactic features it lacks, using techniques described by Alkuhlani et al. (2013).

3.3 Morphological Transformation

The mapping of the morphological features from CATiB++ to NUDAR includes mapping the NUDAR POS tag, as well as the set of features that appear with each token. The mapping of POS tags is done through a lookup that takes the CATiB POS tag and the gold BW tag of the token stem, and maps them to the equivalent NUDAR POS tag. The lookup map is shown in Table 2. The mapping of the morphological features uses another lookup map, that is shown in Table 3.

3.4 Syntactic Transformation

UD and CATiB representations share a number of similarities, both being dependency representations. However there are differences between them that primarily arise from the basic focus on what a dependency is, and affect the structure of the trees. CATiB tries to represent a structure closer to traditional Arabic grammar analysis, which is more interested in modeling the assignment of *case*. This results in function words tending to head their phrase structures more. In contrast, UD tends to get closer to the meaning, and minimize differences between different languages that have different morphosyntactic structures (Nivre, 2016).

The CATiB and NUDAR representations use a different set of labels that refer to very similar concepts, although they use different forms. This results in having a number of similar constructs where we only need to map the labels without modifying the structure.

[4] For more information on this tool, contact the second author.

	DEV		TRAIN		TEST		ALL	
	Tokens	Sentences	Tokens	Sentences	Tokens	Sentences	Tokens	Sentences
PATB1	16,881	447	16,586	487	133,813	3,585	167,280	4,519
PATB2	16,972	264	17,128	228	135,219	2,099	169,319	2,591
PATB3	40,092	1,275	40,411	1,248	321,787	10,105	402,290	12,628
Total	73,945	1,986	74,125	1,963	590,819	15,789	738,889	19,738

Table 1: The tokens and sentences in the current NUDAR Treebank, based on PATB parts 1, 2, and 3

3.4.1 Verbal Constructs

Verbal constructs representation in CATiB and NUDAR are the same, except for the choice of label. The verb heads the optional subject, zero or more objects, and other modifiers. The label used for the attachment between the subject and the verb is `SBJ` in CATiB, and `nsubj` in NUDAR. Any object is attached to the verb using the `OBJ` relation in CATiB. In NUDAR, the first object takes the label `obj`, and any other objects take the label `iobj`. An example of a verbal sentence is demonstrated in Figure 1.

In the case of passive verbs, the subject of the passive verb takes the relation `nsubj:pass`. CATiB marks passive verbs using the POS tag **VRB-PASS**, and uses the relation `SBJ` for the subject.

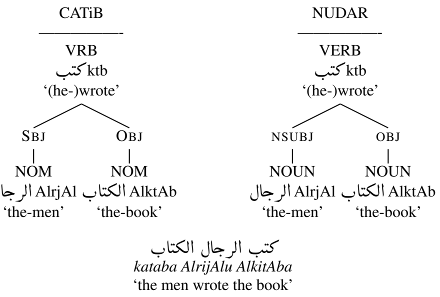

Figure 1: Verb-Subject-Object Construct

3.4.2 Adjectival Constructs

A noun followed by an adjectival modifier maintains the same structure in both CATiB and NUDAR, with the noun heading the adjectival modifier. The label that this relation takes in NUDAR is `amod`, as in the example in Figure 2.

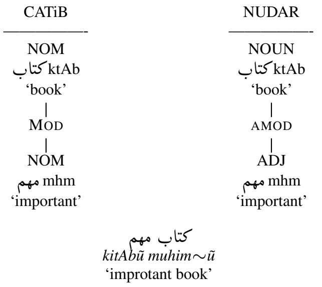

Figure 2: Adjectival Modifier Construct

3.4.3 Idafa Constructs

The *Idafa* construct can be used to mark the genitive possessor, objects of preposition-like nominal adverbs, and some quantification constructs (Habash et al., 2009). Each of these cases is treated differently. For the case of possessive constructs, such as in Figure 3, we extend the existing `nmod` UD label to the language-specific `nmod:poss` label.

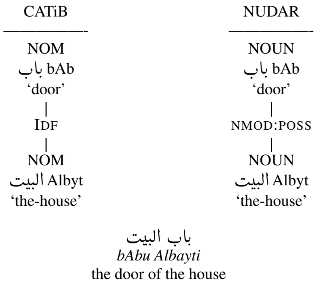

Figure 3: *Idafa* Construct: the genitive possessor

Nominal adverbs (e.g., امام *AmAm* 'front', خلف *xlf* 'behind') are connected to their parents with an *Idafa* relation (`IDF`) in CATiB. Since these nominal-adverbs are tagged with the **ADV** POS tag in NUDAR, this relation gets directly mapped to `advmod`, as demonstracted in Figure 4.

The case of *Idafa* in quantification constructs (Figure 6) will be discussed next with the other number constructs.

3.4.4 Number Constructs

Number constructs take different relational labels in the CATiB representation. A number[5] heads the

[5] The numbers we discuss in this section are three and above. The number *one* in Arabic (واحد *wAHd*) is an ad-

CATiB POS	BW' POS	UD POS
NOM	*SUFF_DO	PRON
NOM	ABBREV	NOUN
NOM	ADJ	ADJ
NOM	ADJ_COMP	ADJ
NOM	ADJ_NUM	ADJ
NOM	ADV	ADV
NOM	DEM_PRON	DET
NOM	DIALECT	X
NOM	FOREIGN	X
NOM	INTERJ	INTJ
NOM	INTERROG_ADV	ADV
NOM	INTERROG_PRON	PRON
NOM	LATIN	X
NOM	NOUN	ADV if *nom-prep*
NOM	NOUN	NOUN if not *nom-prep*
NOM	NOUN_NUM	NUM
NOM	NOUN_PROP	PROPN
NOM	NOUN_QUANT	NOUN
NOM	POSS_PRON	PRON
NOM	PRON	PRON
NOM	REL_ADV	ADV
NOM	REL_PRON	PRON
NOM	TYPO	X
PROP	!!	PROPN
PNX	NUMERIC_COMMA	PUNCT
PNX	PUNC	PUNCT
PRT	CONJ	CCONJ
PRT	CONNEC_PART	PART
PRT	DET	DET
PRT	FOCUS_PART	PART
PRT	FUT_PART	AUX
PRT	INTERJ	INTJ
PRT	INTERROG_PART	PART
PRT	JUS_PART	PART
PRT	NEG_PART	PART
PRT	PART	PART
PRT	PREP	ADP
PRT	PSEUDO_VERB	CCONJ
PRT	RC_PART	PART
PRT	RESTRIC_PART	PART
PRT	SUB_CONJ	SCONJ
PRT	VERB_PART	AUX
PRT	VOC_PART	PART
UNK	DIALECT	X
UNK	LATIN	X
UNK	TYPO	X
VRB	!!	VERB
VRB-PASS	!!	VERB

Table 2: Part-of-Speech mapping from CATiB POS and BW POS to NUDAR. BW' denotes the complete or partial match of the full BW tag set. Entries marked with **!!** under BW' POS means that the relevant information is taken from the CATiB POS tag only. Entries starting with * under the BW means that there are multiple tags the contain this partial tag, and that they all map to the same UD POS. *nom-prep* is a function that determines if the word falls under the list of nominal adverbs, which are specified words that are tagged as nominals in CATiB and PATB, but behave like adverbs.

MADAMIRA	UD
asp	Aspect
cas	Case
stt	Definite
gen	Gender
mod	Mood
num	Number
per	Person
vox	Voice
pos	PronType, NumForm
bw	Polarity

Table 3: Morphological features mapping from CATiB to NUDAR

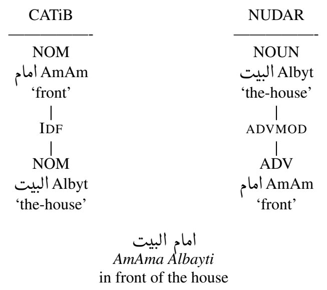

Figure 4: *Idafa* Construct: the object of a preposition-like nominal adverb

word it modifies. The relation between the number and the noun is either *Tamyiz* - if the number is between 11 and 99, as in Figure 5 or *Idafa* otherwise, as in Figure 6. In NUDAR the noun heads the number, and the relation is `nummod` in both cases.

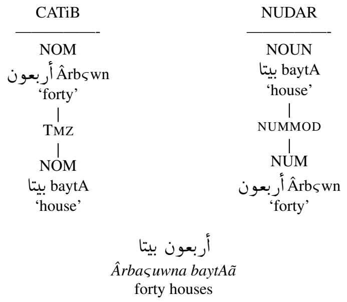

Figure 5: *Tamyiz* Consrtuct: the numeral modifier of numbers between 11 and 99

The relations between numbers in compound number structures in CATiB are similar to the

jective, and will always be headed by the word it modifies. Number *two* (اثنان *AvnAn*) can also be an adjective that attaches low to the word it modifies, or it can be part of the noun's morphology.

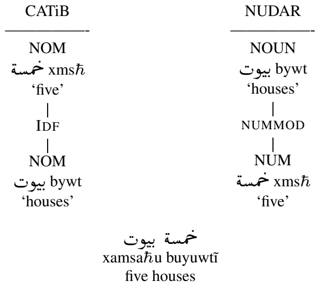

Figure 6: *Idafa* Construct: the numeral modifier

relations between numbers and the nouns they modify. In this example, as shown in Figure 7, الف *Alf* 'thousand' would be headed by أربعون *Ârbςwn* 'forty' with the *Tamyiz* relation (TMZ) in CATiB. However, in NUDAR, the subtree would be headed by *Alf*, and *Ârbςwn* would be attached to it with a compound relation.

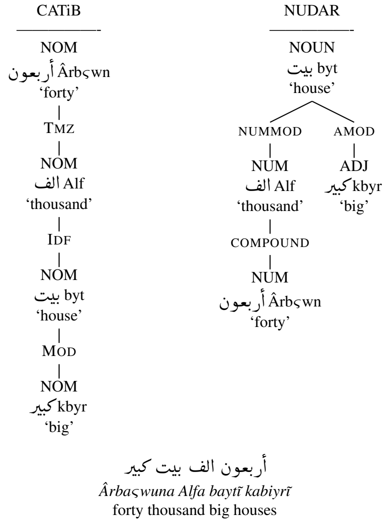

Figure 7: Compound Number Construct

3.4.5 Coordination Constructs

In CATiB, the coordinating conjunction heads the sub-tree of the following phrase in a cascading structure. In NUDAR, however, the construct is flat, with all the coordinating conjunctions and conjuncts being headed by the first conjunct of the coordination construct. The coordinating conjunctions take the relation cc, and the conjuncts take the relation conj. The difference between the two tree structures in illustrated in Figure 8.

It is also common for the coordinating conjunctions in Arabic to be sentence-initial discourse connectives (واو ابتدائية/واو استئنافية) or interruptives (واو اعتراضية) (Habash et al., 2009). In these cases, the coordinating conjunctions are dependent on the root predicate of the sentence with the relation cc.

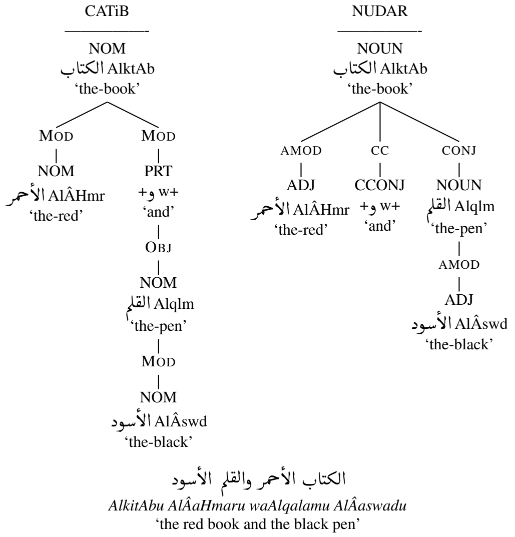

Figure 8: Coordination Construct

3.4.6 Proper Name Constructs

Proper nouns having two or more nominal elements have these elements linked using the language-specific relation flat:name in NUDAR. If a proper noun has more than two nominal elements, they all are headed by the first element of the proper name, unlike CATiB, where each element is headed by the one that precedes it, as seen in Figure 9.

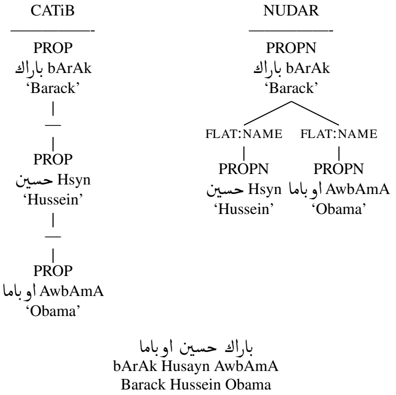

Figure 9: Proper Name Construct

Apposition is marked by the relation appos in NUDAR, as in Figure 10, opposed to the MOD relation it takes in CATiB.

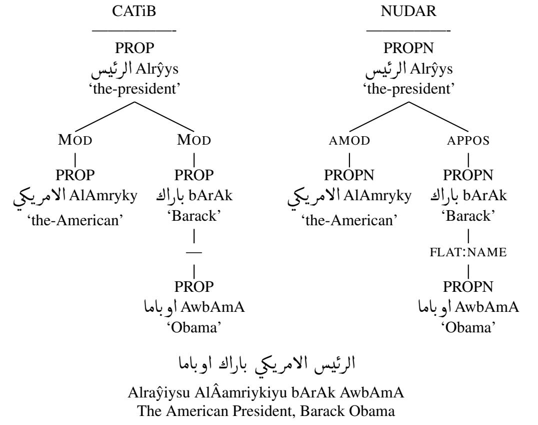

Figure 10: Apposition Construct

3.4.7 Preposition Constructs

Prepositions in NUDAR are case-marking elements that are dependent on the element they introduce. They attach low, unlike the CATiB structure. The label this relation takes is `case`, as can be seen in Figure 11.

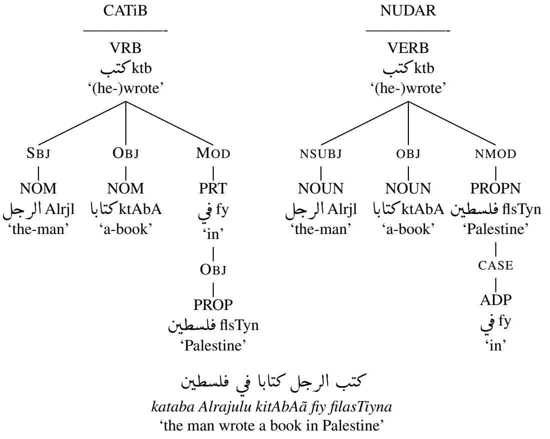

Figure 11: Prepositional Construct

3.4.8 Copular Constructs

The basic copular construct in Arabic does not include copular verbs. It has the same tree structure in both CATiB and NUDAR. The predicate heads the relation, and the subject attaches to it with the label `SBJ` in CATiB and `nsubj` in NUDAR, as seen in Figure 12.

Some so-called *incomplete verbs* in Arabic, such as كان kAn 'to be', and verb-like particles, such as إن Ǎn 'indeed/verily' act like the copula verb *be* in English. Since copula verbs cannot be the heads of clauses, they attach to their predicates with the relation `cop`, like the example in Figure 13.

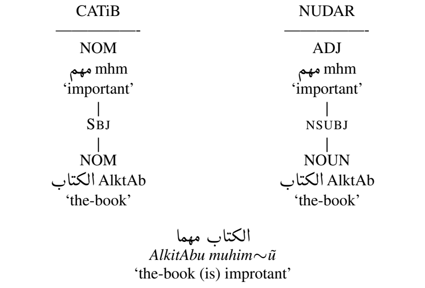

Figure 12: Basic Copular Construct

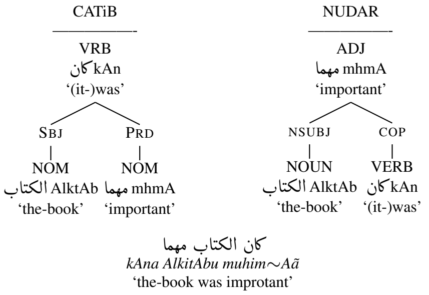

Figure 13: Copular Construct with Copular Verb

3.4.9 Subordinating Conjunction Constructs

Subordinating conjunctions introduce a finite clause that is subordinate to another clause. As with copula, they cannot head a clause. The subordinating clause's predicate becomes the parent of the subordinating conjunction, as shown in Figure 14.

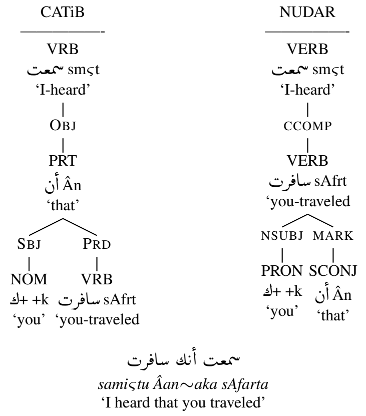

Figure 14: Subordinating Conjunction Construct

3.4.10 Clausal Complement Constructs

Clauses that are a core argument of a verb are attached to that verb with a `ccomp` or an `xcomp` relation. The `ccomp` relation is used for clauses that have their own subject, while `xcomp` refers to clauses with a subject that is the same as the subject of the verb that heads them. An example of a clause attaching with a ccomp relation is in Figure 15.

3.5 Validation

During our conversion process, we selected a random subset of 17 sentences, containing 608 tokens, from the TRAIN set. We manually created a gold reference for this set, and we used it to fine tune our convertor. After we froze the conversion, we converted a randomly selected subset of 82 sentences, containing 2,685 tokens, from the TEST set, that we automatically converted into NUDAR, and manually checked and fixed to create a gold test set. This gold subset was used to test the performance of the final version of the convertor. The scores that we got in both subsets against the gold were very high. We show the Labeled and Unlabeled Attachment Scores (LAS and UAS respectively) and Label Accuracy Score (LAcc) in Table 4.

	LAS	**UAS**	**LAcc**
Dev	98.5%	98.8%	99.3%
Test	98.0%	99.1%	98.3%

Table 4: Conversion scores against manually created gold trees

An error analysis shows that the majority of the errors originated from the gold annotations of the PATB treebank. These errors are caused by either having the wrong dashtag, or attachment in the PATB trees (Habash et al., 2007a). A small number of errors were caused either by bugs in the conversion rules, or by missing rules.

3.6 Comparing PAUDT and NUDAR

A direct comparison between PAUDT and NUDAR proved hard to perform. Even though both treebanks follow the UD guidelines in general, there were many differences originating from the data sources, as well as from the interpretation of the guidelines. The data in PAUDT comes from portions of PATB parts 1 and 2, and from Arabic Gigaword. The data in the current NUDAR treebank comes from PATB parts 1, 2, and 3. In total, NUDAR contains 1,834 documents, and PAUDT contains 874 documents. The two treebanks overlap in 207 documents (based on document IDs). Within these shared documents, we find a number of differences such as PAUDT's inclusion of article titles and full stop sentence segmentation, compared with missing article titles and occasional trees covering multiple sentences in NUDAR. Even among sentences that are similarly segmented, we find many tokenization differences: dates and times are tokenized differently (e.g., *11-5* vs *11 - 5*), as well as specific Arabic words that are treated differently (e.g., فيما *fyma* or في ما *fy mA* 'in that'). Out of the shared 207 documents, only 335 sentences had the same tokenization, and these had additional differences in POS choice as well as tree structures and labels. We plan a more detailed comparison in the future to help consolidate the two treebanks.

4 Parsing Experiment

We conducted some experiments to benchmark the parsing scores in the NUDAR treebank. We also compare the result of parsing directly in NUDAR space to parsing in CATiB space then converting to NUDAR representation.

For our parsing experiments, we used the MaltParser (Nivre et al., 2006) to train an Arabic dependency parser in the space of both CATiB and NUDAR. We compared the output of the NUDAR parser, to the results of converting the output of the CATiB parser to NUDAR using the system described in Section 3.

For the CATiB parser, we used the optimized settings described by Shahrour et al. (2016), and were able to achieve comparable results. We used the gold CATiBex POS tags (Marton et al., 2013), and gold morphological features derived from gold BW tags, to train the parser on the TRAIN dataset of PATB parts 1, 2, and 3. We tested on the TEST dataset of the same treebank parts. The output of the parser was then converted to NUDAR representation.

For the NUDAR parser, we ran the MaltOptimizer (Ballesteros and Nivre, 2012) on the full TRAIN dataset of NUDAR. We used the optimized settings to train and run our parser.

The results of these experiments are shown in Table 5. The first row shows the result of training the MaltParser on the NUDAR training dataset with the optimized settings. The second row shows the results of training the MatlParser on the CATiB training dataset, with the optimized settings from Shahrour et al. (2016). Finally, the last row shows the results of converting the output of the CATiB parser to NUDAR representation.

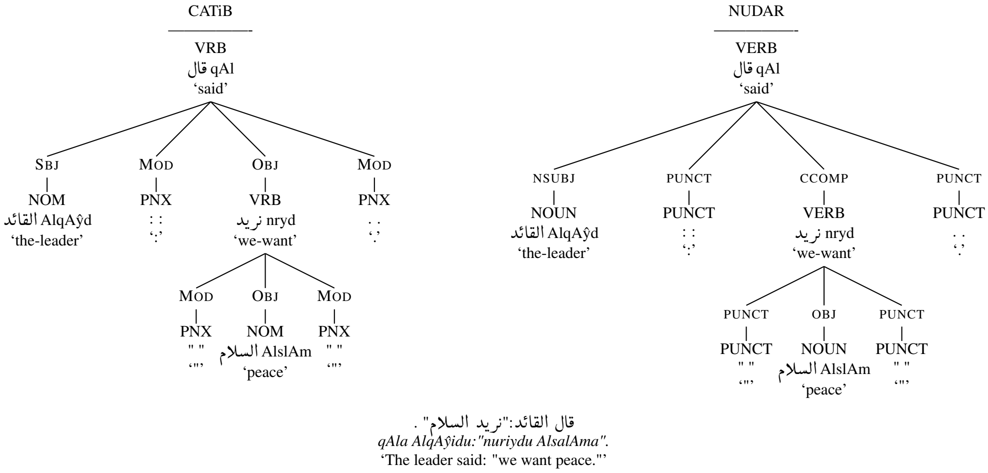

Figure 15: Clausal Complement Construct

Our results show that training a parser in NUDAR space produces better results than training a parser in CATiB space and converting the output to NUDAR representation. This can be attributed to the fact that the output of the CATiB parser does not produce the dashtags that are present in CATiB++, which help in the conversion process.

We also observe that the scores of the NUDAR parser are slightly lower than the scores of the CATiB parser. Although it is not possible to directly compare both parsers because of the different structures, we hypothesize that the larger label set in NUDAR (more than 40 labels compared to the eight labels of CATiB), and factors related to the structures, such as the longer distance between words and their parents in NUDAR (4.4 on average compared to 3.5 in CATiB) may be harder for a parser. We offer these insights as possible explanations, with the assumption that measuring and confirming these hypotheses need more research. It is also possible that further optimization will help increase the scores achieved by the NUDAR parser.

System	**REF**	**LAS**	**UAS**	**LAcc**
NUDAR-Parser	*NUDAR-GOLD*	81.9%	83.7%	93.8%
CATiB-Parser	*CATiB-Gold*	83.1%	85.0%	94.3%
CATiB-Parser+converted	*NUDAR-GOLD*	75.3%	80.0%	88.3%

Table 5: Scores for the NUDAR and CATiB parsing and conversion experiments

5 Conclusion and Future Work

In this paper, we presented a fully automated converter from PATB to the UD syntactic representation. The conversion includes converting the POS tags and other morphological features, as well as the dependency relations and tree structures to UD, through a pipeline of conversion rules. The work was validated through a manually checked test set. We also present the results of an initial parsing experiment. This treebank will be made available as part of the UD v2.0 release as "UD Arabic-NYUAD".

In the future, we plan to improve the conversion process, and to convert the remaining available PATB parts, in both MSA and dialectal Arabic into the UD syntactic representation. We also plan on converting other Arabic dependency treebanks, such as the CATiB treebank, and the Quranic treebank (Dukes and Habash, 2010) into UD. This will require enriching these treebanks with additional morphosyntactic features, as per the techniques described by Alkuhlani et al. (2013). More experiments on optimizing the parsing process are planned, to make use of the available features to improve the parsing results. Finally, we plan on exploiting the NUDAR treebanks and parsers for use in other areas of NLP such as machine translation.

Acknowledgments

The work done by the third author was supported by the grant 15-10472S of the Czech Science Foundation.

References

Željko Agić and Nikola Ljubešić. 2015. Universal dependencies for croatian (that work for serbian, too).

In *The 5th Workshop on Balto-Slavic Natural Language Processing (BSNLP 2015)*.

Sarah Alkuhlani, Nizar Habash, and Ryan Roth. 2013. Automatic morphological enrichment of a morphologically underspecified treebank. In *HLT-NAACL*, pages 460–470.

Héctor Martínez Alonso and Daniel Zeman. 2016. Universal dependencies for the ancora treebanks. *Procesamiento del Lenguaje Natural*, 57:91–98.

Miguel Ballesteros and Joakim Nivre. 2012. Maltoptimizer: an optimization tool for maltparser. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 58–62. Association for Computational Linguistics.

Daša Berović, Željko Agić, and Marko Tadić. 2012. Croatian dependency treebank: Recent development and initial experiments. In *Seventh International Conference on Language Resources and Evaluation (LREC 2012)*.

Tim Buckwalter. 2002. Buckwalter Arabic Morphological Analyzer Version 1.0. Linguistic Data Consortium, University of Pennsylvania, 2002. LDC Catalog No.: LDC2002L49.

Tim Buckwalter. 2004. Buckwalter Arabic Morphological Analyzer Version 2.0. LDC catalog number LDC2004L02, ISBN 1-58563-324-0.

Marie-Catherine De Marneffe and Christopher D. Manning. 2008. The stanford typed dependencies representation. In *Coling 2008: proceedings of the workshop on cross-framework and cross-domain parser evaluation*, pages 1–8. Association for Computational Linguistics.

Marie-Catherine De Marneffe, Bill MacCartney, Christopher D. Manning, et al. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454. Genoa.

Marie-Catherine De Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D. Manning. 2014. Universal stanford dependencies: A cross-linguistic typology. In *LREC*, volume 14, pages 4585–92.

Mona Diab, Nizar Habash, Owen Rambow, and Ryan Roth. 2013. LDC Arabic treebanks and associated corpora: Data divisions manual. *arXiv preprint arXiv:1309.5652*.

Kais Dukes and Nizar Habash. 2010. Morphological Annotation of Quranic Arabic. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, Malta.

Sašo Džeroski, Tomaž Erjavec, Nina Ledinek, Petr Pajas, Zdenek Žabokrtsky, and Andreja Žele. 2006. Towards a slovene dependency treebank. In *Proc. of the Fifth Intern. Conf. on Language Resources and Evaluation (LREC)*.

David Graff, Ke Chen, Junbo Kong, and Kazuaki Maeda. 2006. *Arabic Gigaword*. Linguistic Data Consortium, Philadelphia, second edition.

Nizar Habash and Ryan M. Roth. 2009. CATiB: The Columbia Arabic Treebank. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 221–224. Association for Computational Linguistics.

Nizar Habash, Ryan Gabbard, Owen Rambow, Seth Kulick, and Mitchell P. Marcus. 2007a. Determining case in Arabic: Learning complex linguistic behavior requires complex linguistic features. In *EMNLP-CoNLL*, pages 1084–1092.

Nizar Habash, Abdelhadi Soudi, and Tim Buckwalter. 2007b. On Arabic Transliteration. In A. van den Bosch and A. Soudi, editors, *Arabic Computational Morphology: Knowledge-based and Empirical Methods*. Springer.

Nizar Habash, Reem Faraj, and Ryan Roth. 2009. Syntactic Annotation in the Columbia Arabic Treebank. In *Proceedings of MEDAR International Conference on Arabic Language Resources and Tools*, Cairo, Egypt.

Nizar Habash. 2010. *Introduction to Arabic natural language processing*, volume 3. Morgan & Claypool Publishers.

Jan Hajič, Eva Hajičová, Petr Pajas, Jarmila Panevová, Petr Sgall, and Barbora Vidová-Hladká. 2001. Prague Dependency Treebank 1.0. LDC catalog number LDC2001T10, ISBN 1-58563-212-0.

Jan Hajič, Otakar Smrž, Petr Zemánek, Petr Pajas, Jan Šnaidauf, Emanuel Beška, Jakub Kráčmar, and Kamila Hassanová. 2004. Prague Arabic dependency treebank 1.0. LDC2004T23. web download.

Anders Johannsen, Héctor Martínez Alonso, and Barbara Plank. 2015. Universal dependencies for danish. In *International Workshop on Treebanks and Linguisic Theories (TLT14)*, page 157.

Mohamed Maamouri, Ann Bies, Tim Buckwalter, and Wigdan Mekki. 2004. The Penn Arabic Treebank: Building a Large-Scale Annotated Arabic Corpus. In *NEMLAR Conference on Arabic Language Resources and Tools*, pages 102–109, Cairo, Egypt.

Mohamed Maamouri, Ann Bies, Sondos Krouna, Fatma Gaddeche, and Basma Bouziri, 2009. *Penn Arabic Treebank Guidelines*. Linguistic Data Consortium.

Mohamed Maamouri, Ann Bies, Seth Kulick, Fatma Gaddeche, Wigdan Mekki, Sondos Krouna, Basma Bouziri, and Wadji Zaghouani. 2010a. Arabic treebank: Part 1 v 4.1 ldc2010t13.

Mohamed Maamouri, Ann Bies, Seth Kulick, Sondos Krouna, Fatma Gaddeche, and Wadji Zaghouani. 2010b. Arabic treebank: Part 3 v 3.2 ldc2010t08.

Mohamed Maamouri, Ann Bies, Seth Kulick, Fatma Gaddeche, Wigdan Mekki, Sondos Krouna, Basma Bouziri, and Wadji Zaghouani. 2011. Arabic treebank: Part 2 v 3.1 ldc2011t09.

Mohamed Maamouri, Sondos Krouna, Dalila Tabessi, Nadia Hamrouni, and Nizar Habash. 2012. Egyptian Arabic Morphological Annotation Guidelines.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19(2):313–330, June.

Yuval Marton, Nizar Habash, and Owen Rambow. 2013. Dependency parsing of modern standard Arabic with lexical and inflectional features. *Computational Linguistics*, 39(1):161–194.

Ryan T. McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith B Hall, Slav Petrov, Hao Zhang, Oscar Täckström, et al. 2013. Universal dependency annotation for multilingual parsing. In *ACL (2)*, pages 92–97.

Kadri Muischnek, Kaili Müürisep, Tiina Puolakainen, Eleri Aedmaa, Riin Kirt, and Dage Särg. 2014. Estonian dependency treebank and its annotation scheme. In *Proceedings of 13th Workshop on Treebanks and Linguistic Theories (TLT13)*, pages 285–291.

Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The conll 2007 shared task on dependency parsing. In *Proceedings of the CoNLL shared task session of EMNLP-CoNLL*, pages 915–932. sn.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. MaltParser: A data-driven parser-generator for dependency parsing. In *In Proc. of LREC-2006*, pages 2216–2219.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may. European Language Resources Association (ELRA).

Joakim Nivre. 2014. Universal dependencies for Swedish. In *Proceedings of the Swedish Language Technology Conference (SLTC)*.

Joakim Nivre. 2016. Universal dependencies: Dubious linguistics and crappy parsing? COLING Keynote Talk.

Petya Osenova and Kiril Simov. 2015. Universalizing bultreebank: a linguistic tale about glocalization. *BSNLP 2015*, page 81.

Lilja Øvrelid and Petter Hohle. 2016. Universal dependencies for norwegian.

Arfath Pasha, Mohamed Al-Badrashiny, Mona Diab, Ahmed El Kholy, Ramy Eskander, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan M. Roth. 2014. MADAMIRA: A Fast, Comprehensive Tool for Morphological Analysis and Disambiguation of Arabic. In *Proceedings of the Language Resources and Evaluation Conference (LREC), Reykjavik, Iceland*.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2011. A universal part-of-speech tagset. *arXiv preprint arXiv:1104.2086*.

Sampo Pyysalo, Jenna Kanerva, Anna Missilä, Veronika Laippala, and Filip Ginter. 2015. Universal dependencies for finnish. In *Proceedings of the 20th Nordic Conference of Computational Linguistics, NODALIDA 2015, May 11-13, 2015, Vilnius, Lithuania*, number 109, pages 163–172. Linköping University Electronic Press.

Rudolf Rosa, Jan Mašek, David Mareček, Martin Popel, Daniel Zeman, and Zdeněk Žabokrtský. 2014. HamleDT 2.0: Thirty dependency treebanks stanfordized. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, and Joseph Mariani, editors, *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC 2014)*, pages 2334–2341, Reykjavík, Iceland. European Language Resources Association.

Mojgan Seraji, Filip Ginter, and Joakim Nivre. 2016. Universal dependencies for persian.

Anas Shahrour, Salam Khalifa, Dima Taji, and Nizar Habash. 2016. Camelparser: A system for arabic syntactic analysis and morphological disambiguation.

Otakar Smrž, Jan Šnaidauf, and Petr Zemánek. 2002. Prague Dependency Treebank for Arabic: Multi-Level Annotation of Arabic Corpus. In *Proceedings of the International Symposium on Processing of Arabic*, pages 147–155, Manouba, Tunisia.

Otakar Smrž, Viktor Bielický, Iveta Kouřilová, Jakub Kráčmar, Jan Hajič, and Petr Zemánek. 2008. Prague Arabic dependency treebank: A word on the million words. In *Proceedings of the Workshop on Arabic and Local Languages (LREC 2008)*, pages 16–23, Marrakech, Morocco. European Language Resources Association.

Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Marta Palmer. 2005. The penn chinese treebank: Phrase structure annotation of a large corpus. *Natural language engineering*, 11(02):207–238.

Daniel Zeman. 2008. Reusable tagset conversion using tagset drivers. In *LREC*.