Prague Arabic Dependency Treebank: A Word on the Million Words

Otakar Smrž  Viktor Bielický  Iveta Kouřilová  Jakub Kráčmar  Jan Hajič  Petr Zemánek

Institute of Formal and Applied Linguistics, Charles University in Prague
Malostranské náměstí 25, Prague 1, 118 00, Czech Republic
http://ufal.mff.cuni.cz/padt/  <padt@ufal.mff.cuni.cz>

Abstract

Prague Arabic Dependency Treebank (PADT) consists of refined multi-level linguistic annotations over the language of Modern Written Arabic. The kind of morphological and syntactic information comprised in PADT differs considerably from that of the Penn Arabic Treebank (PATB). This paper overviews the character of PADT and its motivations, and reports on converting and enhancing the PATB data in order to be included into PADT. The merged, rule-checked and revised annotations, which amount to over one million words, as well as the open-source computational tools developed in the project are considered for publication this year.

1. Introduction

Prague Arabic Dependency Treebank (PADT) provides refined linguistic annotations inspired by the Functional Generative Description theory (Sgall et al., 1986; Hajičová and Sgall, 2003) and the Prague Dependency Treebank project (Hajič et al., 2006). The multi-level description scheme discerns functional morphology, analytical dependency syntax, and tectogrammatical representation of linguistic meaning. PADT is maintained by the Institute of Formal and Applied Linguistics, Charles University in Prague. The initial version of PADT (Hajič et al., 2004a) covered over one hundred thousand words of text. PADT was included in the CoNLL 2006 and CoNLL 2007 Shared Task on dependency parsing (Nivre et al., 2007) or in other parsing experiments (Corston-Oliver et al., 2006). The morphological data and methodology of PADT were also used for training automatic taggers (Hajič et al., 2005). PADT is discussed in detail in (Žabokrtský and Smrž, 2003; Hajič et al., 2004b; Smrž, 2007b; Smrž and Hajič, 2008).

Penn Arabic Treebank (PATB) is the largest such resource for Modern Written Arabic that is annotated with structural morphological features, morph-oriented English glosses, and labelled phrase-structure syntactic trees in the predicate-argument style of the Penn Treebank (Marcus et al., 1993). PATB was developed at the Linguistic Data Consortium, University of Pennsylvania, and was published gradually in four major releases (Maamouri et al., 2004a, 2005a,b,c). The source texts are distributed also independently as part of the Arabic Gigaword (Graff, 2007). PADT has been used mostly for the availability of morphological annotations and fully vocalized word forms. Processing the data with machine-learning techniques has resulted in a number of morphological taggers (Habash and Rambow, 2005; Smith et al., 2005; Hajič et al., 2005) and diacritizers (Nelken and Shieber, 2005; Zitouni et al., 2006; Habash and Rambow, 2007). The syntactic information was exploited in particular for parsing (Kulič et al., 2006), grammar extraction (Habash and Rambow, 2004), and automatic case assignment (Habash et al., 2007). The PATB treebank is further described in (Maamouri and Bies, 2004; Buckwalter, 2004a; Maamouri et al., 2004b).

This paper explores the possibility to merge both of these treebanks into a uniform resource that would exceed the existing ones in the level of linguistic detail, accuracy, and quantity. While we advance the PADT style of annotations in this effort, we also largely benefit from the amount of disambiguated information available in PATB.

The new more than one-million-word treebank denoted as PADT 2.0 combines original Prague annotations with the transformed and enhanced Penn data. The preliminary contents of these components are enumerated in Table 1.

2. Functional Morphology

Due to the impact of PATB, the computational linguistics community is well aware of the Buckwalter Arabic Morphological Analyzer (Buckwalter, 2002, 2004b). This system can be characterized as following the lexical-incremental approach to morphology (Stump, 2001), implying that the only clue for discovering a word’s morphosyntactic properties is through its explicit morphs and their supposed prototypical functions.

The functional view of language pursued in PADT requires, on the contrary, an inferential–realizational morphological model capable of more appropriate and deeper generalizations. ElixirFM (Smrž, 2007a,b) is the novel implementation that replaces any earlier functional approximations used in PATB, which were developed also thanks to the Buckwalter analyzer (Smrž and Pajás, 2004). In order to illustrate the differences in the morphological description of PATB versus PADT, let us discuss a few examples. One disambiguated word in the PATB data might offer this information:

Form  $\text{All"Asilokiy"apu}$
Morph  $\text{Al + lAsilokiy" + sp + u}$
Tag  $\text{DET+ADJ+NSUFF\_FEM\_SG+CASE\_DEF\_NOM}$
Gloss  the + wireless / radio + [fem.sg.] + [def.nom.]
Lemma  $\text{[lAsilokiy"\_1]}$
Root  $\text{implicit in the lexicon}$

The entry spells out the full inflected word form in the Buckwalter transliteration, identifies its structure, and describes it with tags and glosses. It also provides the citation form of the lexeme, like its derivational root, might be stored implicitly in the Buckwalter lexicon, but is not readily available in the treebank.

| Form | $\text{All"Asilokiy"apu}$ |
| Morph | $\text{Al + lAsilokiy" + sp + u}$ |
| Tag | $\text{DET+ADJ+NSUFF\_FEM\_SG+CASE\_DEF\_NOM}$ |
| Gloss | the + wireless / radio + [fem.sg.] + [def.nom.] |
| Lemma | $\text{[lAsilokiy"\_1]}$ |
| Root | $\text{implicit in the lexicon}$ |
The corresponding PADT entry by ElixirFM would yield:

Form: \texttt{al-l\-a-silk\-Iyatu}

Morph: \texttt{al >| lA >| FICL |< Iy |< aT |<< "u"}

Tag: \texttt{A-FS1D}

The interpretation is a little more formal. The lexeme of the given grammatical class and meaning is inflected in the parameters expressed by the tag—the adjective is inflected for feminine gender, singular number, nominative case, and definite state. Both the inflected word form and the citation form are explicit in their morphological structure—they are specified via the underlying template of morphs and the inherited root. Merging the template with the root produces the form in the ArabT\textsc{ex} notation, from which the orthographic string or its phonetic version can be generated.

In this very instance, both of the treebank entries seem more equivalent, with both the future marker \texttt{sa-} and the dicative verb, but does handle distinct tokens if the markers are not separated the future marker \texttt{sa-} from an indicative verb, but does handle distinct tokens if the markers are the stand-alone \texttt{sawfa} or \texttt{lan}. Likewise, perhaps unintentionally, \texttt{li-} is not tokenized if followed by a jussive, but it is tokenized if followed by a subjunctive. There is an easy fix to this redundancy if you notice, but why run the risk of it is tokenized if followed by a subjunctive. There is an easy fix to this redundancy if you notice, but why run the risk of it is tokenized if followed by a subjunctive.
Morphological disambiguation of Arabic encompasses sub-problems like tokenization, ‘part-of-speech’ and full morphological tagging, lemmatization, diacritization, restoration of the structural components of words, and combinations thereof. Given a list of possible readings of an input string produced by an analyzer, it can be worthwhile to organize the analyses into a MorphoTrees hierarchy (Smrž and Pajas, 2004) with the string as its root and the full tokens as the leaves, grouped by their lemmas, non-vocalized canonical forms, and partitionings of the string into such forms. The shift from lists to trees enables clearer presentation of the options and their more convenient annotation. MorphoTrees promote gradual focusing on the solution through efficient top-down search or bottom-up pruning using restrictions on the properties of the nodes, and allow inheritance, refinement and sharing of information. MorphoTrees in Figure 2 depict a complex annotation of the string fbn resolved as fa-hum ‘so they’. Alternative ways of annotating and details on the automation of some of the steps in the process are explained in (Smrž, 2007b).

### 3. Dependency Syntax

Morphological annotations identify the textual forms of a discourse lexically and recognize their grammatical properties. The analytical syntactic processing describes the superficial dependency structures in the discourse, whereas the tectogrammatical representation reveals the underlying dependency structures and restores the linguistic meaning. Annotators on the analytical level are represented by dependency trees. Their nodes map, one to one, to the tokens resulting from the morphological analysis and tokenization, and their roots group the nodes according to the division into sentences or paragraphs. Edges in the trees show that there is a syntactic relation between the governor and its dependent. The nature of the government is expressed by the analytical functions of the nodes being linked. Figures 3 and 4 analyze the following sentences:

‘... after he had spent in it almost twenty years.’

في ملتقى الأدب طرحت المجلة قضية اللغة العربية والأخطار التي تهددها.

‘In the section on literature, the magazine presented the issue of the Arabic language and the dangers that threaten it.’

The connection between the PADT dependency analytical trees and the phrase-structure trees of PATB was studied in...
One of the differences between PATB and PADT that must be taken into account when converting the syntactic data is the formal treatment of Arabic compound prepositions, such as bi-sababi ‘because of’, wifqan li- ‘according to’, bi- ‘n-nishbat išā ‘with respect to’. While PATB does not explicitly distinguish this phenomenon, the PADT approach inspired by the dependency formalism of the Prague Dependency Treebank for Czech does provide a formal solution to these and similar cases (Hajič et al., 1999: 162–163). The criteria for regarding multi-word expressions as compound prepositions are that they be well established (usually listed in dictionaries) and not make sense when standing alone without a following nominal phrase. Compound prepositions were gradually gathered during annotations of the PADT analytical syntax, and can as well be extracted from the data. Their lists have been used in the conversion procedure of PATB and in annotation consistency checks. The reattachments of the nodes in the compounds are implemented in the DepDep dependency tree ‘parser’, which matches on nodes in their linear order, but retains, accesses, and possibly modifies their dependency information.

In Figure 3, we present a rule that transforms a SBAR containing a preposition bādā, a conjunction un, and some unspecified other children, in our case the adverbial S clause. This rule would also match on any uninflated preposition followed by vanna. The subroutine calls the recursive ConDep procedure on all the children, attaches the conjunction to the preposition and the rest of the nodes to the conjunction, and assigns the analytical labels AuxP and AuxC to the respective nodes. The preposition with its subtree is then returned. Other SBAR rules would replace the . . . . Of the differences between PATB and PADT that must be integrated into the larger result, allowing more diverse configurations, requiring yet another kind of a translation procedure to be developed and applied on the temporary result. Our implementation of the conversion consists of more consecutive phases as well, but it attempts to use a stronger and more uniform formalism in both the ConDep primary phase and the DepDep secondary phase. Individual conversion rules specify subsequences of nodes and a subroutine that should be triggered if their pattern appears in the data. A rule’s subroutine can manipulate the matching nodes rather freely and can return more than one transformed subtree to the head cannot account for all desired reconfigurations, requiring yet another kind of a translation procedure to be developed and applied on the temporary result. The dotted arc symbolizes that there can be implicit morphosyntactic constraints between the adjacent tokens in the analyses, the consistency of which should be verified.
AuxP of that prepositional phrase bears the auxiliary function AuxY, whereas the complement receives its analytical function according to the role of the whole prepositional phrase in a sentence. If a compound preposition is composed of a noun and two prepositions, e.g. bi-'n-nisbati ilà, the last preposition in the string becomes the head AuxP, and the remaining components depend on it side by side bearing the auxiliary functions AuxY. The complement that follows this compound preposition also hangs on the head AuxP and bears the analytical function determined by the context.

The motivation for this explicit identification of compound prepositions is that on the tectogrammatical layer, these compounds as well as all other synsemantic words disappear. Only the nodes of autosemantic words remain, representing their underlying syntactic and semantic roles.

There are many other kinds of syntactic differences between PATB and PADT. Some structures in PATB tend to be more semantically oriented, while others are rather simplified. Note e.g. the use of QP in Figure 3, which breaks the grammatical dependency between the modifier nahwa, the numeral, and the counted object (cf. Habash et al., 2007). The favored PATB annotation would, however, correspond to the tectogrammatical treatment of quantifiers in PADT. Both ConDep and DepDep tackle even issues due to certain inconsistency of annotations. Flat phrases must be parsed, improper dependencies eliminated, incorrect instances of subordination need to be restructured to coordination, etc. The functional labels in PATB capture several types of adverbials, conflated to Adv on the analytical level. On the other hand, the set of tectogrammatical functors in PADT is yet much more refined (cf. Mikulová et al., 2006).

The complete tree conversion process includes also the resolution of traces. This phase yields pointers of grammatical coreference, present in the manual analytical data as well (Hajič et al., 2004b). The recovery of textual coreference is performed on the tectogrammatical level, cf. Figure 4.
Figure 4: \textbf{Left:} Example of analytical annotation. Orthographic words are tokenized into lexical words, and their inflectional morphosyntactic properties are encoded using the positional tags. Coordination members are depicted with dashed edges. \textbf{Right:} Example of tectogrammatical annotation with resolved coreference (extra arcs) and indicated values of contextual boundness. Lexemes are identified by lemmas, and selected grammatemes are shown in place of morphosyntactic features.

4. \textbf{Tectogrammatics}

Tectogrammatics, the underlying syntax reflecting the linguistic meaning of an utterance, is the highest level of annotation in the family of Prague Dependency Treebanks (Hajič et al., 2006). It captures dependency and valency (Zabokrtský, 2005) with respect to the deep linguistic relations of discourse participants. In its generality, the description also includes topic–focus articulation, coreference resolution, and other non-dependency relations. The set of tectogrammatical annotations in PADT is still rather experimental, yet, we intend to develop this formalism further.

The topology of a tectogrammatical representation of a sentence is similar to that of the analytical level. In contrast to it, nodes in the tree may be deleted, inserted, and even reorganized. We speak of a transfer of structures from analytical to tectogrammatical, which can be partly automated. The nodes appear as lexical entries rather than inflected forms. Grammatemes, the deep grammatical parameters, abstract away from the morphological and analytical features of an utterance. Functors, the deep roles that the participants assume, include Actor, Patient, Addressee, Origin, Effect, various types of local and temporal modifications, Extent, Manner, Cause, Identity, Restriction, coordination types, and many more (Mikulová et al., 2006).

Figure 4 compares the analytical and tectogrammatical representations of a sentence. The inserted nodes are recovered from the discourse as the obligatory actants of the valency frames of the two verbs (cf. Bielický and Smrž, 2008). Values of contextual boundness, a feature from which the topic–focus dichotomy is inferred, are also indicated (cf. Hajičová and Sgall, 2003; Smrž and Hajič, 2008).

There is a number of structures on the analytical level that appear as a co-sign, i.e. their actual meaning results only from being combined. Quite often, one of the nodes in a structure is occupied by a verb while its other members are modifiers, which are not further developed. The tectogrammatical level (t-level) erases some language-specific features present on the analytical level. The treatment of Arabic verbal negation is an instance thereof. Generally, verbal negation on the t-level is expressed by an abstract node for negation. The reason for it being a node and not a grammateme is that the deep ordering of the node with respect to the verb determines the scope of negation, with consequences for the information structure of a discourse.

In Arabic, a variety of combinations, such as \textit{lam yaktub} vs. \textit{mā kataba} ‘he did not write’, turn into exactly the same structures on the t-level. Stylistic variation is not reflected in tectogrammatics—it is believed that \textit{lam yaktub} is more formal, while \textit{mā kataba} is considered rather dialectal or used only in spoken discourse. The \textit{mā} type of negation is used in rather fixed collocations, such as \textit{mā zāla} ‘still’.

The opposition of perfective \textit{kataba} and imperfective \textit{yaktab} forms in Arabic, generally perceived as an opposition of past and non-past, is actually irrelevant for the deep notion of tense in this context. On the analytical level, the tense reference is expressed as a co-sign consisting of a negative particle and a finite verbal form. On the t-level, the tense indication is marked only in the grammatemes of the verbal node, i.e. \texttt{[tense=Ant]} for the anterior tense.

The other markers of verbal negation, \textit{lā} and \textit{lan}, are represented in the same manner, but relate to different tenses. Structurally very similar on the analytical level is the use of a future marker for expressing positive posterior tense. In \textit{sa-yaktabu} ‘he will write’, the modifier \textit{sa-} is attached to the node of the verb, and itself has no offsprings. The more explicit form \textit{sawfa} can also be used. In combination of \textit{sawfa} with negation, the particle \textit{lā} is inserted between the marker and the verb, i.e. \textit{sawfa lā yaktabu}. However, \textit{sa-} cannot combine with negation, in which case \textit{lan} is used instead, as in \textit{lan yaktabu} ‘he will not write’. On the t-level, the modifier node containing the future marker is always deleted and the tense is indicated at the verb \texttt{[tense=Post]}.

An interesting point is the treatment of \textit{gad}, a modifying particle attached to verbs. When connected with the perceptive form of a verb, it has the meaning of an aspectual nuance of completed action, like ‘already’. On the t-level, this
particle is deleted and the verbal node receives the grammaticemes for anteriority and completeness [tense=Ant, aspect=Cpl]. However, when used with an imperfective verbal form, its meaning changes to possibility in the future, ‘it might well be that’. The grammaticemes of the verb become [tense=Post, deontmod=Poss], but the modifier node is retained on the t-level, as is the case with other kinds of modality nodes (cf. Mikulová et al., 2006).

5. Quality Control

Our software environment for maintaining the PADT and PATB data is TrEd, an editor for tree-like structures developed in Perl by Petr Pajas. It is a highly customizable and programmable tool providing both the graphical user interface and the application programming interface used for network-oriented data processing, such as conversions or consistency checks (cf. Štěpánek, 2006). TrEd integrates all the levels of annotation by enabling the user to invoke macros or external programs of any kind. During annotation, one can take great advantage of specific contexts/modes with predefined macro operations, keyboard-shortcuts, and stylesheets for informative display of the data. The dependency approach to syntax does not restrict TrEd itself. Next to MorphoTrees (Smrž and Pajas, 2004), we have for instance implemented contexts for viewing and possibly annotating phrase-structure trees. Most of our quality control programs are written as TrEd scripts. Linguistic and formal constraints on the annotated data, as well as requirements on the mutual compatibility between levels, are implemented with transparent code reusable also in automatic tagging and partial parsing. Using this technology, we have successfully discovered and eliminated many annotation errors and inconsistencies in both PADT and PATB, on all the levels of annotation. We will report on this process in detail in the documentation to PADT 2.0. The order of revisions is in tens of thousands of words, i.e. percents of the whole treebank, which means great improvement in the accuracy of PADT 2.0, and might have significant effect on the performance of any derived applications (cf. Habash et al., 2007). Our experiments on an earlier version of the data indicated that annotating MorphoTrees is up to three times faster than disambiguating morphology in form of the classical MorphoLists. The inter-annotator disagreement on MorphoTrees was 5.3 %, on MorphoLists it reached 9.3 %. The inter-annotator disagreement in the attachment of nodes on the analytical level before revisions was measured to 9.2 %. With the upgrade of the morphological data in the whole treebank to ElixirFM, as well as with the introduction of our new tools for consistency verification and data processing, we believe that the inter-annotator agreement will yet considerably improve on these values.

6. Conclusion

In this contribution, we have overviewed the theoretical concepts behind the Prague Arabic Dependency Treebank, and have discussed converting the Penn Arabic Treebank into the PADT style. We have described the original data and the tools that we develop. PADT 2.0 will be an important new linguistic resource. TrEd with its extensions, ElixirFM, and Encode Arabic are open-source software published under the GNU General Public License:

http://ufal.mff.cuni.cz/~pajas/tred/

http://sf.net/projects/elixir-fm/

http://sf.net/projects/encode-arabic/

Acknowledgements

We would like to thank all our colleagues in the PADT project for their annotation efforts, inspiring ideas and valuable suggestions, and for their long-time support: Tereza Čečáková, Milada Frantová, Petr Pajas, Pavel Pecina, Zdeněk Žabokrtský, and others. This research has been funded by the Ministry of Education of the Czech Republic (MSM0021620838 and MSM0021620823) and by the Grant Agency of the Czech Academy of Sciences (1ET101120413).

References


Nizar Habash, Ryan Gabbard, Owen Rambow, Seth Kulick, and Mitch Marcus. Determining Case in Arabic: Learning Complex Linguistic Behavior Requires Complex Linguistic Fea-