Construct State Modification in the Arabic Treebank

Ryan Gabbard
Department of Computer and Information Science
University of Pennsylvania
gabbard@seas.upenn.edu

Seth Kulick
Linguistic Data Consortium
Institute for Research in Cognitive Science
University of Pennsylvania
skulick@seas.upenn.edu

Abstract

Earlier work in parsing Arabic has speculated that attachment to construct state constructions decreases parsing performance. We make this speculation precise and define the problem of attachment to construct state constructions in the Arabic Treebank. We present the first statistics that quantify the problem. We provide a baseline and the results from a first attempt at a discriminative learning procedure for this task, achieving 80% accuracy.

1 Introduction

Earlier work on parsing the Arabic Treebank (Kulick et al., 2006) noted that prepositional phrase attachment was significantly worse on the Arabic Treebank (ATB) than the English Penn Treebank (PTB) and speculated that this was due to the ubiquitous presence of construct state NPs in the ATB. Construct state NPs, also known as iDAfa[1] (إِضَافَة) constructions, are those in which (roughly) two or more words, usually nouns, are grouped tightly together, often corresponding to what in English would be expressed with a noun-noun compound or a possessive construction (Ryding, 2005)[pp.205–227]. In the ATB these constructions are annotated as a NP headed by a NOUN with an NP complement. (Kulick et al., 2006) noted that this created very different contexts for PP attachment to "base NPs", likely leading to the lower results for PP attachment.

[1]Throughout this paper we use the Buckwalter Arabic transliteration scheme (Buckwalter, 2004).

In this paper we make their speculation precise and define the problem of attachment to construct state constructions in the ATB by extracting out such iDAfa constructions[2] and their modifiers. We provide the first statistics we are aware of that quantify the number and complexity of iDAfas in the ATB and the variety of modifier attachments within them. Additionally, we provide the first baseline for this problem as well as preliminary results from a discriminative learning procedure for the task.

2 The Problem in More Detail

As mentioned above, iDAfa constructions in the ATB are annotated as a NOUN with an NP complement (ATB, 2008). This can also be recursive, in that the NP complement can itself be an iDAfa construction. For example, Figure 1 shows such a complex iDAfa. We refer to an iDAfa of the form `(NP NOUN (NP NOUN))` as a two–level iDAfa, one of the form `(NP NOUN (NP NOUN (NP NOUN)))` as a three–level iDAfa (as in Figure 1), and so on. Modification can take place at any of the NPs in these iDAfas, using the usual adjunction structure, as in Figure 2 (in which the modifier itself contains an iDAfa as the object of the PREP `fiy`).[3]

This annotation of the iDAfa construction has a crucial impact upon the usual problem of PP attachment. Consider first the PP attachment problem for the PTB. The PTB annotation style (Bies et

[2]Throughout the rest of this paper, we will refer for convenience to iDAfa constructions instead of "construct state NPs".

[3]In all these tree examples we leave out the Part of Speech tags to lessen clutter, and likewise for the nonterminal function tags.

```
(NP $awAriE
    [streets]
    (NP madiyn+ap
       [city]
        (NP luwnog  byt$)))
            [Long] [Beach]
```
شَوَارِع مَدِينَة لونغ بيتش

Figure 1: A three level idafa, meaning *the streets of the city of Long Beach*

```
(NP $awAriE
    [streets]
    (NP (NP madiyn+ap
            [city]
            (NP luwnog  byt$))
                [Long] [Beach]
        (PP fiy
            [in]
            (NP wilAy+ap
                [state]
                (NP kAliyfuwrniyA)))))
                    [California]
```
شَوَارِع مَدِينَة لونغ بيتش فِي وِلَايَة كَالِفورنِيَا

Figure 2: Three level iDAfa with modification, meaning *the streets of the city of Long Beach in the state of California*

(A) multi-level PP attachment at same level — allowed

```
(NP (NP ...)
    (PP ....)
    (PP ....))
```

(B) multi-level PP attachment at different levels — not allowed

```
(NP (NP (NP ...)
        (PP ....)
    (PP ....))
```

Figure 3: Multiple PP attachment in the PTB

```
(NP (NP streets)
    (PP of
        (NP (NP the city)
            (PP of (NP Long Beach))
            (PP in (NP the state)
                   (PP of
                     (NP California))))))
```

Figure 4: The English analog of Figure 2

al., 1995) forces multiple PP modifiers of the same NP to be at the same level, disallowing the structure (B) in favor of structure (A) in Figure 3, and parsers can take advantage of this restriction. For example, (Collins, 1999)[pp. 211-12] uses the notion of a "base NP" (roughly, a non–recursive NP, that is, one without an internal NP) to control PP attachment, so that the parser will not mistakenly generate the (B) structure, since it learns that PPs attach to non–recursive, but not recursive, NPs.

Now consider again the PP attachment problem in the ATB. The ATB guidelines also enforce the restriction in Figure 3, so that multiple modifiers of an NP within an iDAfa will be at the same level (e.g., another PP modifier of "the city of Long Beach" in Figure 2 would be at the same level as "in the state..."). However, the iDAfa annotation, independently of this annotation constraint, results in the PP modification of many NPs that are not base NPs, as with the PP modifier "in the state..." in Figure 2. One way to view what is happening here is that Arabic uses the iDAfa construction to express what is often a PP in English. The PTB analog of the troublesome iDAfa with PP attachment in Figure 2 would be the simpler structure in Figure 4, with two PP attachments to the base NP "the city." The PP modifier "of Long Beach" in English becomes part of iDAfa construction in Arabic.

In addition, PPs can modify any level in an iDAfa construction, so there can be modification within an iDAfa of either a recursive or base NP. There can also be modifiers of multiple terms in an iDAfa.[4]

The upshot is that the PP modification is more free in the ATB than in the PTB, and base NPs are no longer adequate to control PP attachment. (Kulick et al., 2006) present data showing that PP attachment to a non–recursive NP is virtually non–existent in the PTB, while it is the 16th most frequent dependency in the ATB, and that the performance of the parser they worked with (the Bikel implementation (Bikel, 2004) of the Collins parser) was significantly lower on PP attachment for the ATB than for PTB.

The data we used was the recently completed revision of 100K words from the ATB3 ANNAHAR corpus (Maamouri et al., 2007). We extracted all oc-

[4]An iDAfa cannot be interrupted by modifiers for non-final terms, meaning that multiple modifiers will be grouped together following the iDAfa. Also, a single adjective can modify a noun within the lowest NP, i.e., inside the base NP.

Number of Modifiers	Percent of iDAfas
1	72.4
2	20.6
3	5.2
4	1.0
5	0.2
8	0.6

Table 1: Number of modifiers per iDAfa

Depth	Percent of Idafas
2	75.5
3	19.9
4	3.8
5	0.8
6	0.1

Table 2: Distribution of depths of iDAfas

currences of `NP` constituents with a `NOUN` or `NOUN`–like head (`NOUN_PROP`, `NUM`, `NOUN_QUANT`) and a NP complement.

This extraction results in 9472 iDAfas of which 3877 of which have modifiers. The average number of idafas per sentence is 3.06.

3 Some Results

In the usual manner, we divided the data into training, development test, and test sections according to an 80/10/10 division. As the work in this paper is preliminary, the test section is not used and all results are from the dev–test section.

By extracting counts from the training section, we obtained some information about the behavior of iDAfas. In Table 1 we see that of iDAfas which have at least one modifier, most (72%) have only one modifier, and a sizable number (21%) have two, while a handful have as many as eight. Almost all iDAfas are of depth three or less (Table 2), with the deepest depth in our training set being six.

Finally, we observe that the distributions of attachment depths of modifiers differs significantly for different depths of iDAfas (table 3). All depths have somewhat of a preference for attachment at the bottom (43% for depth two and 36% for depths three and four), but the top is a much more popular attachment site for depth two idafas (39%) than it is for deeper ones. Level one attachments are very common for depth three iDAfas for reasons which are unclear.

	Depth 2	Depth 3	Depth 4
Level 0	39.0	19.3	16.1
Level 1	17.9	34.8	14.1
Level 2	43.0	9.9	23.6
Level 3		36.0	10.1
Level 4			36.2

Table 3: For each total iDAfa depth, the percentage of attachments at each level. iDAfa depths of five or above are omitted due to the small number of such cases.

Based on these observations, we would expect a simple baseline which attaches at the most common depth to do quite poorly. We confirm this by building a statistical model for iDAfa attachment, which we then use for exploring some features which might be useful for the task, either as a separate post–processing step or within a parser.

To simplify the learning task, we make the independence assumption that all modifier attachments to an iDAfa are independent of one another subject to the constraint that later attachments may not attach deeper than earlier ones. We then model the probabilities of each of these attachments with a maximum entropy model and use a straightforward dynamic programming search to find the most probable assignment to all the attachments together. Formally, we assign each attachment a numerical depth (0 for top, 1 for the position below the top, and so on) and then we find

$$\operatorname*{argmax}_{a_1,\ldots,a_n} \prod_{1}^{n} P(a_1)\ldots P(a_n) \text{ s.t. } \forall x : a_x <= a_{x-1}$$

Our baseline system uses only the depth of the attachment as a feature. We built further systems which used the following bundles of features:

AttSym Adds the part–of–speech tag or non–terminal symbol of the modifier.

Lex Pairs the headword of the modifier with the noun it is modifying.

TotDepth Conjunction of the attachment location, the AttSym feature, and the total depth of the iDAfa.

Features	Accuracy
Base	39.7
Base+AttSym	76.1
Base+Lex	58.4
Base+Lex+AttSym	**79.9**
Base+Lex+AttSym+TotDepth	78.7
Base+Lex+AttSym+GenAgr	79.3

Table 4: Attachment accuracy on development test data for our model trained on various feature bundles.

GenAgr A "full" gender feature consisting of the AttSym feature conjoined with the the pair of the gender and number suffixes of the head of the modifier and the word being modified and a "simple" gender feature which is the same except it omits number.

Results are in table 4. Our most useful feature is clearly AttSym, with Lex also providing significant information. Combining them allows us to achieve 80% accuracy. However, attempts to improve on this by using gender agreement or taking advantage of the differing attachment distributions for different iDAfa depths (3) were ineffective. In the case of gender agreement, it may be ineffective because non–human plurals have feminine singular gender agreement, but there is no annotation for humanness in the ATB.

4 Conclusion

We have presented an initial exploration of the iDAfa attachment problem in Arabic and have presented the first data on iDAfa attachment distributions. We have also demonstrated that a combination of lexical information and the top symbols of modifiers can achieve 80% accuracy on the task.

There is much room for further work here. It is possible a more sophisticated statistical model which eliminates the assumption that modifier attachments are independent of each other and which does global rather than local normalization would be more effective. We also plan to look into adding more features or enhancing existing features (e.g. try to get more effective gender agreement by approximating annotation for humanness). Some constructions, such as the false iDAfa, require more investigation, and we can also expand the range of investigation to include coordination within an iDAfa.

The more general plan is to incorporate this work within a larger Arabic NLP system. This could perhaps be as a phase following a base phrase chunker (Diab, 2007), or after a parser, either correcting or completing the parser output.

Acknowledgments

We thank Mitch Marcus, Ann Bies, Mohamed Maamouri, and the members of the Arabic Treebank project for helpful discussions. This work was supported in part under the GALE program of the Defense Advanced Research Projects Agency, Contract Nos. HR0011-06-C-0022 and HR0011-06-1-0003. The content of this paper does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

References

ATB. 2008. Arabic Treebank Morphological and Syntactic guidelines. http://projects.ldc.upenn.edu/ArabicTreebank.

Ann Bies, Mark Ferguson, Karen Karz, and Robert MacIntyre. 1995. Bracketing guidelines for Treebank II-style Penn Treebank project. Technical report, University of Pennsylvania.

Daniel M. Bikel. 2004. Intricacies of Collins' parsing model. *Computational Linguistics*, 30(4).

Tim Buckwalter. 2004. Arabic morphological analyzer version 2.0. LDC2004L02. Linguistic Data Consortium.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, Department of Computer and Information Sciences, University of Pennsylvania.

Mona Diab. 2007. Improved Arabic base phrase chunking with a new enriched pos tag set. In *Proceedings of the 2007 Workshop on Computational Approaches to Semitic Languages*.

Seth Kulick, Ryan Gabbard, and Mitchell Marcus. 2006. Parsing the Arabic Treebank: Analysis and improvements. In *Proceedings of TLT 2006*. Treebanks and Linguistic Theories.

Mohamed Maamouri, Ann Bies, Seth Kulick, Fatma Gadeche, and Wigdan Mekki. 2007. Arabic treebank 3(a) - v2.6. LDC2007E65. Linguistic Data Consortium.

Karin C. Ryding. 2005. *A Reference Grammar of Modern Standard Arabic*. Cambridge University Press.