Parsing the Arabic Treebank: Analysis and Improvements

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

Ryan Gabbard Mitchell Marcus
Department of
Computer and Information Science
University of Pennsylvania
{gabbard,mitch}@cis.upenn.edu*

1 Introduction

Previous work has demonstrated that the performance of current parsers on Arabic is far below their performance on English or even Chinese, which in turn harms performance on NLP tasks that use parsing as an input. This paper is an exploration of some of the issues involved in this difference. We focus on the Collins parsing model [3] as implemented in the Bikel parser [1]. The corpus used for the experiments is the Arabic Treebank [6] (ATB).

We cluster these issues in three ways. First, it is important when comparing Arabic parsing performance to other languages that the comparison be a fair one; therefore we first discuss some issues around evaluation and show that current Arabic parsing performance is not quite as bad as previously thought. Second, we present some modifications to the parser which provide modest increases in performance. Finally, we explore deeper differences between the Arabic Treebank and the Penn Treebank and advance some speculations as to why parsers have difficulty with Arabic.

2 Background to Parsing Experiments

The first results on parsing the ATB were presented in [1]. This work was based on data available early in the ATB project, from what is now ATB1. The results are

*We would like to thank Ann Bies, Dan Bikel, Tim Buckwalter, Mark Liberman, Mohamed Maamouri, and Wigdan Mekki for many useful discussions. This work was supported in part by an NSF graduate fellowship, and in part under the GALE program of the Defense Advanced Research Projects Agency, Contract Nos. HR0011-06-C-0022 and HR0011-06-1-0003. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of DARPA.

Hajič, J. and Nivre, J. (eds.): Proceedings of the TLT 2006, pp. 31–42.

Run	<= 40	All	Description
1	75.4/76.0	72.5/73.4	taken from Bikel dissertation
2	75.15/76.26	73.11/74.19	on current ATB1 files

Figure 1: Summary of Parsing Results on ATB1

shown as "Run 1" in Figure 1, and are separated into scores for all sentences and those of length <= 40, as is commonly done. This was a small training and test set, consisting of approximately 149K tokens for training and 11K for testing. The parser used was Dan Bikel's implementation of the Collins parser [3].[1]

An immediate problem for this early work was the interaction between the parser and the ATB tag set. Since a tag is the result of a morphological analysis represented by the different morphemes being concatenated with a plus sign, the tagset is very large. Such a large tagset is problematic for the current version of the parser, in which the tags are treated as indivisible units, which is a reasonable enough assumption for the Penn Treebank (PTB) but not for parsing the ATB. In addition, as Bikel points out, it is not even a static tag set since a new tag can be "created" from a combination of different cliticized forms that had not appeared in the training data.

For these two reasons, a mapping from the ATB tags to a much smaller set of 20 PTB tags was developed by Ann Bies and is now included in ATB releases.[2]

Both the unvoweled and voweled forms of the corpus were available for parsing, but the decision was made to use the unvoweled data, "because that would ultimately be necessary for any real-world parser." At the same time the gold part-of-speech tags were used (although reduced) "because the purpose was for bootstrapping treebank annotation."

Ideally, parser evaluation would be in the context of using a POS tagger to produce the tags for the parser input, instead of using the gold tags. It is also not necessary to assume that a "real-world" parser will only have unvoweled data, since a preprocessing step could restore some of the diacritics.

However, despite our concerns over these points, we continue with the basic set-up that Bikel initiated because we are concerned with evaluating improvements from his baseline and because the main current utility for the parser is as a preprocessing step for ongoing ATB annotation, where the input data that has gone through a level of POS annotation that results in gold tags and tokenized data. Also, as described in Section 3.6, we are currently making changes to the POS tagset and wish to wait until this is finalized before using a POS tagger for the input. We also continue to use the unvoweled data.[3]

[1] The Bikel parser is available at `http://www.cis.upenn.edu/~dbikel/software.html`.

[2] It was of course realized that such a reduced tagset is throwing away much potentially useful information, and we present data in Section 3.4 showing that this is the case. However, it was an attempt to get the parser working at some minimal level of functionality.

[3] See [7] for some experimentation using various versions of the voweled data.

Corpus	Train	Test
ATB1	148,825	11,100
ATB3	314,811	18,837
PTB	950,028	56,684
PTB Comparable	314,828	56,684

Table 1: Token count for ATB1, ATB3, PTB, and PTB with a comparable size training set

	Recall/Precision
With capitalization	87.46/87.63
Without capitalization	87.42/87.72

Table 2: English parsing results for a model trained on the comparably–sized corpus.

The first step in our experiments was to duplicate Bikel's results. Since Bikel's work, revised versions of ATB1 have been released with various annotation corrections and a more extensive tag set with case and imperfect verb mood endings (with a correspondingly revised tagset reduction). We used these updated versions of the same files that Bikel had used for his experiments, along with the current version of the Bikel parser (0.9.9c). The results are shown as Row 2 in Figure 1. The score for sentences $<= 40$ is very close, but for "all" it is somewhat higher. We are not sure why this is, but assume it is because of improvements to the ATB1 annotation or the parser since the original work was done.

For the rest of the work in this paper, we used a different training/test set. We decided to use ATB3, since it is the most recent section of the ATB and has benefited the most from experience in annotation. We also felt that ATB3 is large enough that it could be used for experimentation without complicating evaluation by including material from the other sections of the ATB, which might differ in various ways.

To construct our experimental corpus, we first concatenated all of ATB 3. Then, following Bikel's methodology for creating his training/test set, we removed all trees rooted in the nonterminal `X`, on the assumption that those sentences have problematic annotation, eliminating 759 trees. Then, of every 10 trees, we used 8 for training, 1 for development, and 1 for test. All the results reported here are for the development set. Table 1 summarizes the sizes of the training and test sets for ATB1 and ATB3.[4]

Before moving on to the parsing results, we recap the state of PTB parsing. A fair comparison of parser performance on Arabic compared to English requires that we use an English parsing model trained on a comparably–sized corpus. We generated such a corpus by random sampling from the WSJ corpus, as indicated in Table 1. The parsing results are summarized in Table 2. To be completely fair, we also tested an entirely lowercase version of English (since Arabic orthography has no case), which resulted in a negligibly different parsing score.

[4]For ATB3, "test" in the table refers to the development set. The ATB3 training, test, and development set total approximately 350K tokens. The ATB3 documentation states that there are 400,213 tokens after clitic separation, with the difference due to the 759 trees we eliminated.

Run	<= 40	All	Description
3	72.28/73.95	68.26/70.19	Baseline
4	74.93/77.13	70.06/72.78	Improved handling of punctuation
5	75.56/78.10	71.13/74.26	Fix of verb–intervening condition
6	77.31/79.44	73.10/75.19	Improved tagset mapping
7	77.41/79.63	73.15/75.22	Treating NP as argument
8	78.14/80.26	73.61/75.64	Fix for headless VPs

Figure 2: Summary of Recall/Precision Parsing Results on ATB3

3 Improving the Parser

Table 2 is a summary of the improvements to the parser. In the following subsections we describe the details of each run.

3.1 Run 3 - Baseline

The first step was to run the baseline version of the parser as described above on the new training/development set. As can be seen, comparing Run 2 (Figure 1) and Run 3 (Figure 2), there is a significant drop in the performance on sentences <=40 and an even more significant loss when all sentences are included. It is not clear to us why this is so. One might think that the loss on sentences of length > 40 is because sentences are longer in ATB3. There is some difference in the average sentence length of our test data for ATB3 (32.06 tokens) compared to ATB1 (28.61), but it seems unlikely that that accounts for the score difference, although both are significantly longer than in the PTB test set (23.46).[5] The reason for the lower performance on the shorter sentences is also unclear. In any case, we take the results in Run 3 as our current baseline, and focus on improvements from this starting point.

3.2 Run 4 - Improved Handling of Punctuation

The annotation of punctuation in the Penn Treebank and the Arabic Treebank differs in certain ways that have a significant impact on the score, as can be seen by comparing runs 3 and 4.[6]

The Penn Treebank uses a variety of labels for punctuation symbols. The period tag is used for the preterminals `(. .)` `(. ?)` `(. !)`, and the colon tag is used for the preterminals `(: :)` `(: -)` `(: ;)`. The double apostrophe

[5] We have been told anecdotally that there are more sentences in ATB3 consisting of multiple Ss as sisters to a root S with no punctuation separating them, which would likely account for the loss for longer sentences, but we have not yet verified this.

[6] We apologize in advance for discussing punctuation in such detail. However, this is an example of where little-discussed assumptions in parsing and evaluation can be important, and part of our goals is to get such assumptions into the open.

is used as a tag for itself: (" "), and likewise for the slanted double apostrophe (` `), comma (, ,), and left and right parentheses (-LRB- -LRB-) (-RRB- -RRB-).

Parsing evaluation is standardly done using the evalb program with a parameter file (included with the Bikel parser) that customizes it in various ways. One such parameterization is to ignore tokens with certain labels for purposes of evaluation. The labels that are ignored are: { , : ` ` . }.

In contrast, the ATB uses the label PUNC to cover all of these punctuation tags, along with a few others either not present in the PTB or given the tag SYM there. And, crucially, double quotations are represented by a double quotation mark instead of two apostrophes: (PUNC "). As discussed above, the Bikel parser does a mapping of the ATB tags to a reduced set. As part of this, it maps (PUNC ,) to (, ,), although other tokens with tag PUNC are left as is.[7]

A consequence of this difference in the treatment of punctuation in the corpus and in evaluation is that the evaluation for PTB parsing ignores tokens with labels , : ` ` . but the evaluation for the ATB ignores only tokens with the comma label. The solution to this issue, as done in run 4, is obvious: modify the treatment of punctuation in the ATB to be as similar as possible to that in the PTB. We extend the internal mapping to also map (PUNC .) to (. .), (PUNC ?) to (. ?), and so on. We have kept the ATB usage of the real double-quotation mark, and so map (PUNC ") to (" "). We have modified the parameter file for evalb in exactly one way, to include " as a label to ignore, so that quotation marks are ignored in the ATB just as they are in the PTB.

The difference in the score between Run 3 and Run 4 is almost entirely due to ignoring quotation marks for the evaluation, which can often appear inside or outside the brackets to which they belong. The resulting difference in the score highlights how careful one must be when comparing results of different parsing setups, as even small items of evaluation can have a significant impact.[8]

However, we emphasize that in our view there is no principled reason for eliminating quotation marks, or indeed any punctuation, from the parser evaluation for Arabic. The parser is currently being used as a preparsing step for ongoing treebank annotation, and if a punctuation token is in the wrong location, it needs to be fixed up by the treebanker just as much as any other misplaced token. But for purposes of comparison to parsing results for the PTB, it is appropriate to ignore such punctuation.[9]

[7] This is done in the file danbikel/parser/arabic/TagMap.java included in the Bikel distribution. The Bikel distribution contains an additional parameter file for evalb, BIKEL.prm, which modifies the COLLINS.prm file to contain additional labels to delete, particular PU. However, there is no such tag in the ATB; it is probably leftover from an early version of the the ATB.

[8] The modification of punctuation also has an impact on the internal workings of the parser, as far as what tokens get pruned and then reinserted before and after parsing. We leave aside discussion of this since space is limited and this pruning or lack of does not seem to have an major impact.

[9] The likely reason for ignoring such labels in the PTB evaluation was that quotation marks were known to be too inconsistent to expect the parser to get them correct.

3.3 Run 5 - Fixing the Verb–Intervening Condition

Both the Collins parser and the Bikel re–implementation condition modifier attachment on various items in the history of the phrase being constructed. In particular, when attaching a modifier, one of the history conditions is a boolean indicating whether a verb was contained in the yield of the previous modifiers, allowing the parser to learn a preference for lower attachment. Due to space considerations, we refer the reader to [3] for further discussion.

The important point here is that in the Bikel release of the Arabic language package, this conditioning is broken. It determines if a token is a verb by checking if its pos tag begins with `VERB`. However this refers to the tags *before* they have been reduced (in the early version of the ATB), while the parser is only working with the reduced tag set. Therefore, the verb-intervening condition is always false, depriving the parser of some significant information.

We fixed this problem by encoding that the verb tags consist of VBP, VBN, VBD, VB, thus allowing the verb–intervening condition to work properly. As can be seen comparing Run 4 and Run 5, this resulted in a further improvement.

3.4 Run 6 - Improved Tagset Mapping

The tagset reduction distributed with the ATB is quite extreme, in that all prefixes and suffixes are eliminated. For example,
`NOUN+NSUFF_FEM_SG+CASE_DEF_ACC` and
`DET+NOUN+NSUFF_FEM_SG+CASE_DEF_ACC` are both mapped to `NN`.[10]

As mentioned in footnote 2, it is a common assumption that important information is being lost by such a reduction, although without significant modification the parser is currently unable to take advantage of detailed morphological information in POS tags, since the tags are treated as indivisible items. However, we have found that even without a more extensive modification to the parser, a significant gain can be obtained just by lessening the severity of the tagset reduction. In particular, we made two modifications to the reduction:

1. We kept the determiner prefix information. For example,
   `NOUN+NSUFF_FEM_SG+CASE_DEF_ACC` continues to be mapped to `NN`, while
   `DET+NOUN+NSUFF_FEM_SG+CASE_DEF_ACC` is mapped to `DT+NN`.

2. Demonstrative tags are mapped to a new tag `DEM` instead of being collapsed with `DT`. Previously `DEM_PRON_F` (and all other variations of `DEM_PRON`) was mapped to `DT`, and `DET` was also mapped to `DT`. Under our modification, `DEM_PRON_*` is mapped to `DEM` while `DET` is mapped to `DT`.

It seemed likely that throwing away the determiner information was particularly bad, and it is a pleasant result that retaining it does give a nice increase to

[10] Some of the number information is incorporated into the tags. For example, `NOUN+NSUFF+FEM_PL` is mapped to `NNS`, where the latter is the tag for a plural noun.

```
(NP (NOUN+CASE_DEF_GEN <iTAr )
     framework/context+[def.gen.]
    (NP (NP (NOUN+CASE_DEF_GEN juhuwd)
              efforts+[def.gen.])
            (NP (NOUN_PROP wA$inoTun))
                  Washington
        (PP (PREP li)
                 in
            (NP (NP (NOUN+NSUFF_FEM_SG+CASE_DEF_GEN mkAfHp)
                     confrontation/battle
                    (NP (DET+NOUN+CASE_DEF_GEN AlArhAb)))
                        the terrorism/terrorizing+[def.gen]
                (PP (PREP fy)
                          in
                    (NP (DET+NOUN+CASE_DEF_GEN AlEAlm))))))
                         the world
```

Figure 3: An example of the construct state.

the score, even though the number of tags has increased and there is some data fragmentation in the parsing model of nouns depending on whether they have determiners.[11] Further analysis is needed to determine exactly where the parser is able to take advantage of this additional information.

3.5 Run 7 - Treating NP as argument

One aspect of phrase structure in the ATB that is not present in the PTB is the "construct state" (Idafa construction), which has the effect of nouns taking NP complements. This construction roughly corresponds to possession constructions and noun compounds in English and is very common in the ATB. One example is shown in Figure 3.[12]

We hypothesize that the existence of this construction is responsible in some deep ways for decreased parsing performance, as discussed in Section 4, although here we limit ourselves to a slight change in the parser configuration. The specification of what are considered arguments for a head with some parent is part of the "language package" for customizing the parser, and for Arabic includes such conditions as that any SBAR, S, or VP child of a VP parent is an argument. We have added the condition that any NP child of an NP is considered an argument. (Except for adjunction structures, which in general are excluded from argument annotation.) This had a very negligible effect on the parsing, but it seems to be conceptually correct, and so we have left it in.[13]

[11] This fragmentation was before, and still is, present in that two tokens that differ only in whether one has a determiner are treated as two completely different tokens by the parser. But with the original tagset reduction they were collapsed under the same pos tag, whereas now that is not the case.

[12] All Arabic examples use the transliteration system in [2].

[13] It seems likely that one reason for the negligible effect on parsing is that it is probably subsumed

```
(VP (NOUN+NSUFF_FEM_SG+CASE_DEF_GEN mgAdrt)
                                    departure
    (NP-SBJ (POSS_PRON_3MS h))
                          his/its
    (NP-OBJ (DET+NOUN+CASE_DEF_ACC Albyt)
                                the house
            (DET+ADJ+CASE_DEF_ACC AlAbyD)))))
                                the white
```

Figure 4: A VP with no verbal head. (“his departing the White House”)

3.6 Run 8 - Fix for Headless VPs

Inspection of the parsing results and the ATB has revealed a significant number of inconsistencies with the Part of Speech tags that we suspect are having a harmful effect on the parsing accuracy. We discuss here one such case, and in Section 4 we refer to some future work along these lines.

Approximately 5% of the VPs in ATB have a head that is not verbal. An example is shown in Figure 4. The categorization of the heads of such VPs is a complex topic for Arabic annotation (and is currently being revised for ATB annotation) which we cannot discuss further here. However, to evaluate the effect of such nonverbal heads, we modified our training and test corpus so that all nouns and adjectives that were heads of a VP were modified to have a new POS tag, `DV` (deverbal). In addition, there were many cases of `lys` (meaning “not”) tagged as a `NEG_PART` heading a VP. Following advice of the ATB annotators, we modified such instances of `lys` to have a `PV` (perfect verb) tag.

These two changes lead to some further improvement for the parsing results. It is possible that the use of the `DV` tag, while easing the job of the parser, puts an additional burden on the POS tagger, due to the increased pos ambiguity of lexical items. Still, this seems to us a prime area for further exploration, including examining the heads of other phrasal projections, as part of a general cleanup of the POS tags in the treebank, as discussed some in Section 4.1.

4 Analysis of Parser Errors and Future Work

Breaking down parsing scores by dependency type is a useful way to gain insight into the parsing performance. A few interesting results of doing this are highlighted in Table 3 and discussed in the subsections below; these form the basis for our future work.[14]

under the “distance” condition, as explained in [3].

[14]This dependency analysis is based on that presented in [3], although it is our own implementation.

Dependency	A%	E%	ARk	ERk	AR	AP	AF	ER	EP	EF
Base NP										
NPB TAG TAG L	12.4	30.7	2	1	79.0	78.5	78.7	95.6	93.2	94.4
NPB TAG TAG R	6.0	0.5	6	24	84.0	83.3	83.7	74.6	79.7	77.1
Subjects and Objects										
VP TAG NP R	9.2	5.6	4	4	85.5	85.8	85.7	90.6	91.1	90.9
S VP NP L	2.1	7.7	9	3	87.7	81.3	84.4	95.4	95.1	95.3
PP–attachment										
VP TAG PP R	6.3	4.1	5	7	84.1	76.1	79.9	83.0	78.35	80.6
NP NPB PP R	5.3	5.3	7	6	74.9	69.3	72.0	85.7	84.3	85.0
NP NP PP R	1.1	0.1	16	87	26.6	50.00	34.7	0.0	0.0	0.0
Internal Structure of NPs										
NP NPB NP R	1.3	0.5	15	25	50.8	56.0	53.3	71.4	72.4	71.9
NP NP NP R	0.8	0.0	18	97	56.8	65.7	60.9	5.9	16.7	8.7
NP TAG NP R	9.9	0.0	3		85.4	83.2	84.3			
English Score Inflation?										
VP TAG VP R	0.3	5.5	37	5	95.0	80.9	87.4	98.0	98.0	98.0

Table 3: Highlights of a comparative dependency analysis of English and Arabic. A stands for Arabic, E for English, % for the percentage of all dependencies which are of that type, Rk for the frequency rank of that dependency, R for recall, P for precision, and F for F–measure. Read 'NP NPB PP R' as 'NP headed by NPB attaching a PP on the right;' TAG indicates any preterminal symbol.

4.1 Base Noun Phrases

A "Base Noun Phrase" (NPB) is roughly an NP that does not itself dominate an NP. They are given a special status in the parser, as discussed in detail in [1].

A major difference between ATB and PTB parsing is the performance on the base NPs. The dominant case of an NPB with its head on the right (`NPB TAG TAG L`) attaching a preterminal symbol to its left accounts for 30.7% of all dependencies in English and are parsed with an excellent F–measure of 94.4. In Arabic, they are less frequent, although still quite significant (12.4%), but are far more poorly parsed (78.7%). The significant difference in performance on base noun phrases suggests that investigation of base NPs should be a high priority, especially since errors on base noun phrases can percolate up to errors in surrounding brackets.

One strong possibility is that the POS tag set (the ATB POS tags, not the parser's reduced tags) is too coarse in some cases to adequately capture the distributional pattens of the items that make up a base NP. For example, only certain words can appear before the head noun, roughly quantifiers and some titles. However, these words are tagged as `NOUN`, creating a bias towards allowing inappropriate multiple `NOUN`s in a base NP.[15]

[15] Since the base NPs are treated by the parser with a Markovian structure, this most likely adds to the problem, although space prohibits discussion here.

	Arabic		English	
	Gold %	Test %	Gold %	Test %
VP	50.8	53.5	44.3	45.2
NP	**8.8**	**4.3**	2.7	1.9
NPB	39.7	41.7	52.7	52.9
Overgen.	**4.2**		1.2	

Table 4: PP attachment in the gold standard and parsed data for English and Arabic.

4.2 Prepositional Phrase Attachment and the Internal Structure of Noun Phrases

Another major source of difference in parsing performance is prepositional phrase attachment (12.7% of Arabic dependencies, 9.5% of English). While English and Arabic have similar performance on attaching prepositional phrases to VPs, Arabic is much worse on attaching prepositional phrases within NPs. (72.0% compared to 85.0%).

We strongly suspect that a reason for this is the ubiquitous presence of the construct state NPs in the ATB, in which an NP is headed by a noun taking a NP complement. By definition of a base NP, the parent NP is not a base NP. However, a PP can adjoin to the parent, non-base NP, as in Figure 3 (the NPs headed by `juhuwd` and `mkAfHp` are both non-base NPs to which a PP is adjoined). This is in contrast to the PTB, in which the NP receiving a PP adjunct is virtually always a base NP. In the ATB, the NP receiving a PP adjunct may be a base NP or a non-base NP.

This has two effects. First, it introduces a recursive rule ("NP goes to an NP head with PP adjunct" in addition to the "NP goes to base NP with PP adjunct" rule in both the PTB and ATB) into the grammar with a significant number of occurrences; [5] has documented the detrimental effects of such recursive rules on the probability models of generative parsers. Second, simply by learning that it is possible to attach a PP to an NP or an NPB, a PP now has more options for places to attach, whereas in the PTB it was more often a straight "verb or noun" choice.

The problems that plague PP attachment to NPs are most likely also present for NP attachment to NPs, as can be seen in the "Internal Structure of NPs" section of Table 3, in particular the entry for `NP NPB NP R`, which is only 53.3% for Arabic as compared to 71.9% for English.

Table 4 shows another perspective on this problem, detailing the greater range of attachment possibilities for Arabic as compared to English. Note that also PP–attachment dependencies to NP and VP are overgenerated more often in Arabic, suggesting PPs are getting stolen from other types of constituents.[16]

Table 5 compares the divergences between the frequencies of different non–terminal symbols between gold standard and parsed data in both English and Ara-

[16] One possibility is that they are being taken from `ADJPs`, which contain other nonterminal symbols, such as PP, more commonly than in the PTB.

Arabic		English	
Cat	Div	Cat	Div
ADJP	-10.8%	ADJP	-12.9%
ADVP	-22.6%	ADVP	-16.5%
NP	**-15.2**%	NP	**-4.0**%
PP	0%	PP	0%
S	-5.8%	S	-5.8%
SBAR	-1.9%	SBAR	-11%
VP	-1.7%	VP	0%
WHNP	-37.1%	WHNP	-21%

Table 5: Divergences in nonterminal frequency between gold standard and parsed data. Percentages are changes in frequencies in the parsed data relative to the gold standard.

Sentence Type	Arabic %	English %
VSO	62	0
SVO	17	90
No VP	19	11
Subjectless VP	2	0

Table 6: Varieties of sentence structure in Arabic and English.

bic. It is interesting to note that for both English and Arabic (and for all categories, not just the common ones shown, excepting English `QP`s) the parsed data always undergenerates nonterminals compared to the gold standard. We hypothesize that this may indicate a bias of generative parsers in favor of less structured output (since more structure means more probabilities to multiply together). For Arabic in particular, significant differences are seen for ADVP, NP, SBAR, and WHNP. Most significant here is NP due to its high frequency in the corpus. It is possible a significant number of errors in parsing NPs (and PP attachment) arise not only from attachment errors but from failing to generate parts of the structure entirely.

4.3 Sentence Structure and Verbal Subcategorization

The `VP TAG NP R` and `S VP NP L` dependencies highlight another difference between English and Arabic: the degree of variance in sentence structure. English sentence structure is almost invariably SVO; Arabic, on the other hand, demonstrates both a base VSO word order and a topicalized SVO order, as well as an increased incidence of sentences with predicative NPs and ADJPs in place of verb phrases (Table 6).

The greater variance in sentence structure in Arabic has several negative effects since the subcategorization models are independent both of material on the other side of the head and of material below the current head, preventing the left and right sides of the verb from 'communicating.' We intend to rework the subcategorization system to fix this and related problems.

4.4 Score 'Inflation'

While it will not improve the Arabic score, it is worth noting that the score for English is somewhat 'inflated' by the presence of numerous VPs of the form `(VP aux VP)` which are very easy to parse, with a 98.0 F–measure. The closest Arabic analogue is the `NP TAG NP R` dependency arising from the construct state; however, parsing performance for these is at best mediocre (84.3 F–measure). The original PARSEVAL measure avoided this problem to some extent by, among other things, deleting auxiliaries; this again emphasizes, as pointed out by [4], that the PARSEVAL metric is very different from that used by `evalb`.

5 Conclusion

We have shown that it is important to be sensitive to apparently minor details of the evaluation when comparing parsing performance across languages; in particular, we have shown that parsing performance in Arabic is somewhat better than was previously believed. We have presented several modifications for achieving a modest improvement in Arabic parsing accuracy, and have focused attention on those outstanding issues we believe are crucial for further improvement.

References

[1] Daniel M. Bikel. *On the Parameter Space of Lexicalized Statistical Parsing Models*. PhD thesis, Department of Computer and Information Sciences, University of Pennsylvania, 2004.

[2] Tim Buckwalter. Arabic morphological analyzer version 1.0. Linguistic Data Consortium catalog number LDC2002L49,ISBN 1-58563-257-0, 2002.

[3] Michael Collins. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29:589–637, 2003.

[4] Ryan Gabbard, Seth Kulick, and Mitchell Marcus. Fully parsing the Penn Treebank. In *HLT-NAACL*, pages 184–191, 2006.

[5] Mark Johnson. PCFG models of linguistic tree representations. *Computational Linguistics*, 24:612–632, 1998.

[6] Mohamed Maamouri, Ann Bies, Tim Buckwalter, and Wigdan Mekki. The Penn Arabic Treebank: Building a large-scale annotated Arabic corpus. In *NEMLAR International Conference on Arabic Language Resources and Tools*, pages 102–109, 2004.

[7] Mohamed Maamouri, Ann Bies, and Seth Kulick. Diacritization: A challenge to Arabic treebank annotation and parsing. In *Proceedings of the British Computer Society Arabic NLP/MT Conference*, London, UK, October 2006.