Going to the Roots of Dependency Parsing

Miguel Ballesteros[*]
Complutense University of Madrid

Joakim Nivre[**]
Uppsala University

Dependency trees used in syntactic parsing often include a root node representing a dummy word prefixed or suffixed to the sentence, a device that is generally considered a mere technical convenience and is tacitly assumed to have no impact on empirical results. We demonstrate that this assumption is false and that the accuracy of data-driven dependency parsers can in fact be sensitive to the existence and placement of the dummy root node. In particular, we show that a greedy, left-to-right, arc-eager transition-based parser consistently performs worse when the dummy root node is placed at the beginning of the sentence – following the current convention in data-driven dependency parsing – than when it is placed at the end or omitted completely. Control experiments with an arc-standard transition-based parser and an arc-factored graph-based parser reveal no consistent preferences but nevertheless exhibit considerable variation in results depending on root placement. We conclude that the treatment of dummy root nodes in data-driven dependency parsing is an underestimated source of variation in experiments and may also be a parameter worth tuning for some parsers.

1. Introduction

It is a lesson learned in many studies on natural language processing that choosing the right linguistic representation can be crucial for obtaining high accuracy on a given task. In constituency-based parsing, for example, adding or deleting nodes in syntactic trees can have a substantial impact on the performance of a statistical parser. In dependency parsing, the syntactic representations used offer less opportunity for transformation, given that the nodes of a dependency tree are basically determined by the tokens of the input sentence, except for the possible addition of a dummy word acting as the root of the tree. In this article, however, we show that even this seemingly trivial modification can make a difference, and that the exact placement of the dummy root node can have a significant impact on the accuracy of a given parser. This suggests that the placement of the dummy root is a parameter worth tuning for certain parsing systems as well as a source of variation to be taken into account when interpreting experimental results.

2. Dependency Graphs

Dependency-based approaches to syntactic parsing assume that the syntactic structure of a sentence can be analyzed in terms of binary dependency relations between lexical

[*] Universidad Complutense de Madrid, Departamento de Ingeniería del Software e Inteligencia Artificial, C/ Prof. José García Santesmases, s/n, 28040 Madrid, Spain. E-mail: miballes@fdi.ucm.es.

[**] Uppsala University, Department of Linguistics and Philology, Box 635, SE-75126 Uppsala, Sweden. E-mail: joakim.nivre@lingfil.uu.se.

Submission received: 25 July 2012; revised submission received: 13 October 2012; accepted for publication: 19 October 2012

units, units that in the simplest case are taken to correspond directly to the tokens of the sentence. It is very natural to represent this structure by a directed graph, with nodes representing input tokens and arcs representing dependency relations. In addition, we can add labels to arcs in order to distinguish different dependency types or grammatical functions, such as subject, object, adverbial, etc. We call such a graph a *dependency graph*.

Dependency graphs are normally assumed to satisfy certain formal constraints, such as the single-head constraint, which forbids more than one incoming arc to a node, and the acyclicity constraint, ruling out cyclic graphs. Many dependency theories and annotation schemes further require that the graph should be a tree, with a unique root token on which all other tokens are transitively dependent, while other frameworks allow more than one token to be a root in the sense of not having any incoming arc. A simple and elegant way of reconciling such cross-framework differences and arrive at a single formalization of dependency structures is to add a *dummy root node*, a special node that does not correspond to any input token, and to require that the dependency graph is a tree rooted at this node. The original tree constraint can then be enforced by requiring that the special node has exactly one child, but not all frameworks need to enforce this constraint. An additional advantage of adding a dummy root node is that its outgoing arcs can be labeled to indicate the functional status of what would otherwise simply be unlabeled root nodes. With a slight misuse of terminology, we call such labels *informative root labels*.[1]

Since the dummy root node does not correspond to an input token, it has no well-defined position in the node sequence defined by the word order of a sentence and could in principle be inserted anywhere (or nowhere at all). One option that can be found in the literature is to insert it at the end of this sequence, but the more common convention in contemporary research on dependency parsing is to insert it at the beginning, hence treating it as a dummy word prefixed to the sentence. This is also the choice implicitly assumed in the CoNLL data format, used in the CoNLL shared tasks on dependency parsing in 2006 and 2007 (Buchholz and Marsi 2006; Nivre et al. 2007) and the current de facto standard for exchange of dependency annotated data.

The question that concerns us here is whether the use of a dummy root node is just a harmless technicality permitting us to treat different dependency theories uniformly, and whether its placement in the input sequence is purely arbitrary, or whether both of these choices may in fact have an impact on the parsing accuracy that can be achieved with a given parsing model. In order to investigate this question empirically, we define three different types of dependency graphs that differ only with respect to the existence and placement of the dummy root node:

1. **None:** Only nodes corresponding to tokens are included in the graph.
2. **First:** A dummy root node is added as the first token in the sentence.
3. **Last:** A dummy root node is added as the last token in the sentence.

Figure 1 illustrates the three types of dependency graphs with examples taken from the Penn Treebank of English (Marcus, Santorini, and Marcinkiewicz 1993), converted to dependency structure using the procedure described in Nivre (2006), and the Prague Dependency Treebank of Czech (Hajič et al. 2001; Böhmová et al. 2003). In the former case, it is assumed that the dummy root node always has exactly one child, with a

[1] For example, in the Prague Dependency Treebank, which allows multiple children of the dummy root node, the label may indicate whether the child functions as a main predicate, as the head of a coordinate structure, or as final punctuation.

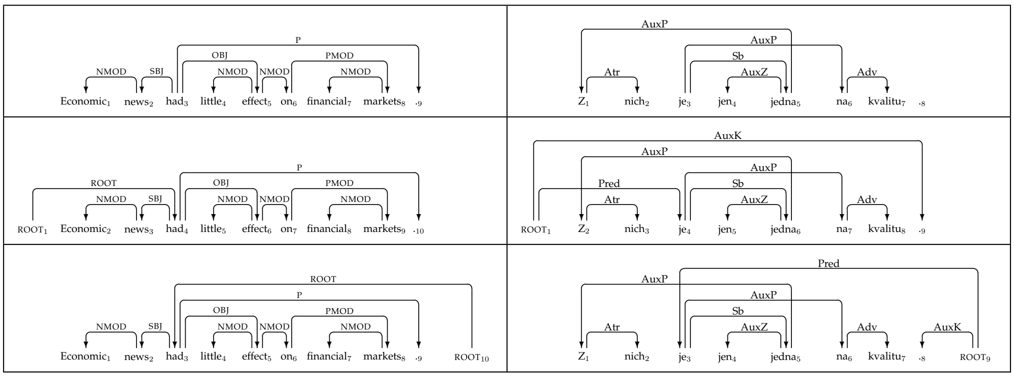
P
OBJ
PMOD
NMOD
SBJ
NMOD
NMOD
NMOD
Economic$_1$ news$_2$ had$_3$ little$_4$ effect$_5$ on$_6$ financial$_7$ markets$_8$.$_9$
AuxP
AuxP
Sb
Atr
AuxZ
Adv
Z$_1$ nich$_2$ je$_3$ jen$_4$ jedna$_5$ na$_6$ kvalitu$_7$.$_8$
P
ROOT
OBJ
PMOD
NMOD
SBJ
NMOD
NMOD
NMOD
ROOT$_1$ Economic$_2$ news$_3$ had$_4$ little$_5$ effect$_6$ on$_7$ financial$_8$ markets$_9$.$_{10}$
AuxK
AuxP
AuxP
Pred
Sb
Atr
AuxZ
Adv
ROOT$_1$ Z$_2$ nich$_3$ je$_4$ jen$_5$ jedna$_6$ na$_7$ kvalitu$_8$.$_9$
ROOT
P
OBJ
PMOD
NMOD
SBJ
NMOD
NMOD
NMOD
Economic$_1$ news$_2$ had$_3$ little$_4$ effect$_5$ on$_6$ financial$_7$ markets$_8$.$_9$ ROOT$_{10}$
Pred
AuxP
AuxP
Sb
Atr
AuxZ
Adv
AuxK
Z$_1$ nich$_2$ je$_3$ jen$_4$ jedna$_5$ na$_6$ kvalitu$_7$.$_8$ ROOT$_9$

Figure 1
Dependency graph types None (top), First (middle) and Last (bottom) for an English sentence from the Penn Treebank (left) and a Czech sentence taken from the Prague Dependency Treebank (right). (Gloss of Czech sentence: Z/Out-of nich/them je/is jen/only jedna/one-FEM-SG na/to kvalitu/quality ./. = “Only one of them concerns quality.”)

dummy dependency label ROOT. In the latter case, the dummy root node may have several children and these children have informative root labels indicating their function (Pred and AuxK in the example). Note also that the Czech dependency graph of type None is not a tree, but a forest, since it consists of two disjoint trees.

3. Experiments

In order to test the hypothesis that the existence and placement of the dummy root node can have an impact on parsing accuracy, we performed an experiment using two widely used data-driven dependency parsers, MaltParser (Nivre, Hall, and Nilsson 2006) and MSTParser (McDonald 2006), and all the 13 data sets from the CoNLL 2006 shared task on multilingual dependency parsing (Buchholz and Marsi 2006) as well as the English Penn Treebank converted to Stanford dependencies (de Marneffe, MacCartney, and Manning 2006). We created three different versions of each data set, corresponding to the representation types None, First and Last, and used them to evaluate MaltParser with two different transition systems – arc-eager (Nivre 2003) and arc-standard (Nivre 2004) – and MSTParser with the arc-factored non-projective algorithm (McDonald et al. 2005). The results are shown in Table 1.

When creating the data sets, we took the original version from the CoNLL-X shared task as None, because it does not include the dummy root node as an explicit input token. In this representation, the tokens of a sentence are indexed from 1 to n and the dependency graph is specified by giving each word a head index ranging from 0 to n, where 0 signifies that the token is not a dependent on any other token in the sentence. The First version was created by adding an extra token at the beginning of the sentence with index 1 and head index 0, increasing all other token and head indices by 1, meaning that all tokens that previously had a head index of 0 would now be attached to the new dummy root token. The Last version was created by adding an extra token at the end of the sentence with index n+1, and changing every head index that previously was 0 to n+1. In both First and Last, we made sure that the new dummy token had a unique word form and unique values for all other features, so that it could not be mistaken for any real word. For First and Last, we applied an inverse transformation to the parser output before evaluation.

Both MaltParser and MSTParser by default add a dummy root node at the beginning of the sentence internally before parsing, so we had to modify the parsers so that they only considered arcs involving nodes corresponding to input tokens. For MaltParser this only required setting a flag that makes the parser start with an empty stack instead of a stack containing an extra dummy root node.[2] For MSTParser, we modified the parser implementation so that it extracts a maximum spanning tree that is still rooted in an extra dummy root node but where the score of a tree is based only on the scores of arcs connecting real token nodes. Finally, since MaltParser with the arc-eager and arc-standard transition systems can only construct projective dependency graphs, we projectivized all training sets before training the MaltParser models using the baseline pseudo-projective transformation of Nivre and Nilsson (2005).[3] Except for these modifications, all parsers were run with out-of-the-box settings.

3.1 Deterministic Arc-Eager Parsing

The arc-eager transition-based parser first described in Nivre (2003) parses a sentence in a single pass from left to right, using a stack to store partially processed tokens and greedily choosing the highest-scoring parsing action at each point. The arc-eager property entails that every arc in the output graph is added at the earliest possible opportunity, which means that right-dependents are attached to their head before they have found their own right-dependents. This can be an advantage because the early attachment neither implies nor precludes the later addition of right-dependents, but it can also be a drawback because it forces the parser to make an early commitment about right-dependents. In this context, it is especially relevant that the addition of a dummy root node at the beginning of the sentence (First) forces the parser to make an early commitment regarding dependents of this root node. By contrast, if a dummy root node is added at the end of a sentence (Last), decisions regarding root dependents will be postponed until the end. Similarly, if no root node is added (None), then these decisions will not be explicitly modeled at all, meaning that whatever nodes remain on the stack after parsing will be treated as root dependents.

As can be seen from Table 1, the arc-eager parser performs consistently worse under the First condition, with an average unlabeled attachment score (UAS) of 83.67 over the 14 languages, to be compared with 84.35 for None and Last. The difference in accuracy between First and None/Last ranges from 0.10 for Dutch to 1.72/1.78 for Slovene, and the difference in means is highly statistically significant ($p < 0.001$, Wilcoxon signed-rank test). The difference between None and Last is never greater than 0.20 (and very far from statistically significant), indicating that either postponing or excluding root attachment decisions leads to very similar performance for the arc-eager parser. The same pattern is found for labeled attachment score (LAS), but here we can only directly compare First and Last because the Arabic, Czech, Portuguese and Slovene data sets contain informative root labels that cannot be predicted under the None condition (cf. footnote 2). The difference in means between First and Last is 0.80 and again highly statistically significant ($p < 0.001$, Wilcoxon signed-rank test). A closer look at the root

[2] The exact commandline flag is `-allow_root false`. A side effect of this flag is that all unattached tokens get the dummy label ROOT, meaning that informative root labels cannot be predicted for representations of type None. See http://maltparser.org for more information.

[3] This is done with the MaltParser flag `-pp baseline`.

		AE				AS				MST			
Language	Type	LAS	UAS	RR	RP	LAS	UAS	RR	RP	LAS	UAS	RR	RP
Arabic	None	*60.00*	**75.17**	74.24	69.52	*60.48*	77.29	81.69	80.60	**66.73**	**78.96**	**84.07**	83.78
	First	63.63	74.57	**84.75**	**73.75**	64.93	77.09	**83.73**	**81.79**	66.41	78.32	83.39	**90.44**
	Last	**64.15**	74.97	73.56	68.24	**65.29**	**77.31**	82.71	81.06	66.41	78.32	78.31	87.83
Bulgarian	None	**85.76**	**90.80**	**94.22**	**90.80**	85.06	90.33	**91.96**	**91.96**	86.30	**91.64**	**98.24**	**98.24**
	First	84.64	89.83	90.20	87.56	85.12	90.33	91.71	91.71	**86.32**	91.28	97.49	97.49
	Last	**85.76**	90.78	**94.22**	90.58	**85.16**	90.33	**91.96**	**91.96**	86.14	91.28	97.24	97.24
Chinese	None	85.13	89.68	**93.63**	88.42	**85.25**	90.08	**93.06**	**93.06**	**86.88**	**90.82**	**94.33**	**94.33**
	First	84.59	89.09	92.25	**89.55**	85.23	**90.10**	92.82	92.82	86.54	90.68	**94.33**	**94.33**
	Last	**85.15**	**89.70**	**93.63**	88.42	85.17	90.00	92.82	92.82	86.36	90.52	93.87	93.97
Czech	None	*68.30*	81.14	80.51	74.61	*68.36*	81.96	**87.85**	73.35	76.70	85.98	82.20	80.83
	First	72.98	79.96	**83.33**	72.66	**74.88**	**82.52**	86.44	**88.95**	77.04	86.34	85.88	84.92
	Last	**73.96**	**81.16**	81.07	**75.53**	74.28	81.78	87.01	73.16	**77.68**	**86.70**	**89.55**	**89.55**
Danish	None	82.36	87.88	91.33	88.06	81.64	**87.86**	92.88	**93.17**	83.39	89.46	92.57	91.44
	First	80.60	86.59	86.69	82.11	**81.66**	**87.86**	92.88	**93.17**	**83.97**	**89.84**	**94.74**	**94.94**
	Last	**82.38**	**87.94**	**91.64**	**88.36**	81.52	87.74	92.88	92.31	83.43	89.42	92.26	92.26
Dutch	None	**71.09**	**74.51**	65.56	66.08	70.67	74.43	**69.84**	72.53	**79.05**	**83.49**	**79.77**	79.92
	First	70.81	74.41	**72.18**	57.88	**71.07**	**75.23**	64.20	**82.09**	78.91	83.43	74.32	83.59
	Last	71.05	**74.51**	65.76	**66.67**	70.65	74.45	**69.84**	72.38	78.25	82.95	75.10	**85.21**
English	None	**88.63**	**90.47**	**88.70**	81.75	88.15	**90.07**	87.83	86.42	87.55	89.91	87.70	87.70
	First	88.00	90.04	83.99	**85.42**	88.04	89.95	86.20	86.24	87.63	90.00	**90.20**	**90.17**
	Last	88.57	90.46	88.58	81.73	**88.16**	**90.07**	**87.91**	**86.61**	**87.69**	**90.06**	88.45	88.38
German	None	83.85	86.64	**94.68**	85.14	84.31	87.22	93.84	93.84	85.64	89.54	**97.76**	**97.76**
	First	83.29	86.08	89.64	**90.40**	**84.37**	**87.24**	93.84	93.84	**85.74**	**89.66**	97.48	97.48
	Last	**83.93**	**86.72**	**94.68**	85.14	84.35	87.22	93.84	93.84	85.34	89.50	**97.76**	**97.76**
Japanese	None	**89.85**	92.10	92.74	85.20	**90.15**	**92.30**	92.53	85.42	90.45	93.02	**93.38**	88.47
	First	88.79	91.27	88.15	**89.20**	89.13	91.57	87.83	**90.94**	**90.83**	**93.36**	92.85	**91.58**
	Last	89.77	**92.12**	**92.96**	84.56	90.01	92.28	**92.64**	85.60	90.47	93.06	92.21	90.66
Portuguese	None	*79.12*	88.60	**92.01**	85.76	*78.32*	87.78	90.28	90.28	84.87	89.74	89.58	89.58
	First	83.77	88.36	87.85	**86.35**	83.45	**87.82**	**90.62**	**90.62**	**85.19**	**90.26**	**91.67**	**91.67**
	Last	**84.17**	**88.62**	**92.01**	85.76	**83.47**	87.80	**90.62**	**90.62**	84.89	89.26	90.62	90.31
Spanish	None	**78.64**	**82.51**	83.25	76.28	**77.88**	**81.69**	**81.73**	**81.73**	79.40	**83.57**	79.70	78.50
	First	78.14	82.15	79.70	73.36	77.64	81.51	81.22	81.22	79.20	83.41	83.76	**84.18**
	Last	**78.64**	82.49	**83.76**	**76.39**	77.72	81.55	80.71	81.12	**79.48**	83.53	**84.77**	83.50
Swedish	None	83.49	89.60	**93.32**	**90.07**	**82.65**	**89.42**	**92.03**	91.56	81.36	88.29	89.97	90.21
	First	83.13	89.29	91.77	89.47	82.53	89.38	91.77	**91.77**	**81.76**	**88.59**	91.52	91.75
	Last	**83.59**	**89.70**	**93.32**	**90.07**	**82.65**	89.36	91.77	91.30	81.66	88.35	**92.03**	**92.03**
Slovene	None	*64.25*	79.56	73.98	63.46	*63.67*	79.28	**75.26**	67.35	71.44	82.47	79.08	75.98
	First	67.73	77.84	71.94	**64.83**	69.40	**79.42**	73.47	**78.69**	**71.72**	**82.67**	**79.34**	**79.34**
	Last	**69.98**	**79.62**	**75.00**	64.05	**69.42**	79.28	**75.26**	67.66	71.64	82.33	76.79	79.21
Turkish	None	**56.66**	**72.18**	**90.29**	92.25	**57.00**	72.06	**90.59**	92.13	58.49	74.55	**93.47**	86.88
	First	56.48	71.86	88.77	**93.00**	56.80	**72.12**	89.68	**94.86**	58.59	74.59	92.56	94.28
	Last	56.64	72.16	90.14	91.95	56.88	72.10	**90.59**	92.13	**58.89**	**74.83**	93.02	**94.45**
Average	None	*76.94*	**84.35**	86.32	81.24	*76.69*	84.41	**87.24**	85.24	79.88	86.53	88.70	87.40
	First	77.61	83.67	85.09	81.11	78.16	**84.44**	86.17	**88.48**	**79.99**	**86.60**	**89.25**	**90.44**
	Last	**78.41**	**84.35**	**86.45**	**81.25**	**78.20**	84.38	87.18	85.18	79.88	86.44	88.71	90.17

Table 1
Experimental results for arc-eager (AE), arc-standard (AS) and maximum spanning tree parsing (MST) on all the CoNLL-X data sets plus the English Penn Treebank converted to Stanford dependencies with three different dependency graph types (None, First, Last). Evaluation metrics are labeled attachment score (LAS), unlabeled attachment score (UAS), root attachment (or no attachment in the case of None) measured as recall (RR) and precision (RP). Scores in bold are best in their column (per language); scores in italic are not comparable to the rest because of informative arc labels that cannot be predicted with the None representation.

accuracy suggests that most of the difference stems from a lower recall on root dependents with the First representation, but this pattern is not completely consistent across languages and Arabic, Czech and Dutch actually have higher recall. For Czech and Dutch this is accompanied by lower precision, but for Arabic the First representation actually gives the best recall and precision of root dependents (but nevertheless the worst overall attachment score). It is probably significant that the Arabic data set has the longest sentences with the root word often appearing early in the sentence. Hence, an early commitment to root attachment is more likely to be correct in this case, even if it is more error prone in general.

3.2 Deterministic Arc-Standard Parsing

The arc-standard transition-based parser first described in Nivre (2004) is similar to the arc-eager parser in that it parses a sentence in a single pass from left to right, using a stack to store partially processed tokens and greedily choosing the highest-scoring parsing action at each point. However, it differs by postponing the attachment of right-dependents until the complete subtree under the dependent itself has been built. As a consequence, the dependency tree is built strictly bottom-up, which means that attachment to a dummy root node will always happen at the end, regardless of whether the dummy root node is positioned at the beginning or at the end of the sentence. There is therefore no reason to expect that the placement of the dummy root node should have the same impact as for the arc-eager parser.

Looking at the results for the arc-standard parser in Table 1 confirms this expectation, with the three conditions giving very similar mean UAS – 84.41 for None, 84.44 for First, 84.38 for Last – and none of the differences being statistically significant. For LAS, we can again only directly compare First and Last, but there is practically no difference in the means here either – 78.16 for First versus 78.20 for Last. Nevertheless, it is worth noting that, for individual languages, differences in scores can be quite substantial. Thus, for Dutch, the First condition outperforms the None/Last condition by 0.80/0.78 in UAS and 0.40/0.42 in LAS. Conversely, for Japanese, the None/Last conditions are better than First by 0.73/0.71 in UAS and 1.02/0.88 in LAS. Although the general trend is that None and Last give the most similar results, just as for the arc-eager parser, there are also cases like Chinese where None and First are both slightly better than Last. Zooming in on root accuracy, we see a clear gain in precision (and marginal drop in recall) for the First representation, which is probably an effect of the arc-standard strategy where the attachment of right-dependents often have to be delayed while left-dependents can be attached eagerly.

3.3 Maximum Spanning Tree Parsing

The maximum spanning tree parser described in McDonald et al. (2005) uses a very different parsing model, compared to the two transition-based parsers. Instead of scoring individual parsing actions, it scores all possible dependency arcs in the sentence and then uses exact inference to extract the highest-scoring complete dependency tree under an arc-factored model, where the score of each tree is the sum of the scores of its component arcs. Since the parsing algorithm does not impose any ordering at all on different attachments, we would expect even less impact from the placement of the dummy root node than for the deterministic arc-standard parser.

The results in Table 1 do not quite confirm this expectation. The mean UAS varies from 86.44 for the Last condition to 86.60 for the First condition, while the mean LAS is 79.88 for None and Last but 79.99 for First. Although none of these differences is statistically significant on the aggregate level, differences can be quite substantial for individual languages, with First outperforming Last by a whole percentage point in UAS for Portuguese (but only 0.30 in LAS) and None outperforming both First and Last by 0.64 for Arabic (and 0.32 in LAS). The fact that LAS differences tend to be smaller than UAS differences can probably be explained by the fact that MSTParser uses a two-stage approach, where the second labeling stage is the same for all three conditions. With respect to root accuracy, the most interesting observation is that both First and Last seem to given higher precision than None, which suggests that it is an advantage to represent root attachments explicitly so that features over these arcs can contribute to the overall score of a parse tree. It is also worth noting that these features are different for First and Last, despite the arc-factored model, because of the so-called "in-between features" that record the part-of-speech tags of words occuring between the head and the dependent of an arc, which in turn explains why these two conditions do not always give the same results.

4. Discussion

The main conclusion we draw from this experiment is that the addition of a dummy word prefixed or suffixed to a sentence is not a mere technical convenience without impact on empirical results. Whether we include a dummy word representing the root of the dependency tree and, if so, where we place this word in the sequence of input tokens can have a non-negligible effect on parsing accuracy for different parsers, in some cases resulting in statistically significant differences with a magnitude of several percentage points according to standard evaluation metrics.

The nature and magnitude of the impact definitely depends on the parsing model used. While the deterministic arc-eager parser gives consistently worse results with a dummy root node positioned at the beginning of the sentence, neither the deterministic arc-standard parser nor the maximum spanning tree parser has any clear preference in this respect. However, although the overall patterns emerging when averaging over many data sets can largely be explained in this way, there is also considerable variation across data sets that we do not yet fully understand. Zooming in on root accuracy has allowed us to start forming hypotheses, such as the impact of long sentences in combination with predominantly head-initial structures for Arabic, but a full exploration of the interaction of parsing models and language-specific properties is clearly outside the scope of this article and has to be left for future research. Another limitation of the current study is that it only examines three different parsers, and while this is clearly sufficient to prove the existence of the phenomenon it will be interesting to see whether the same patterns can be found if we examine more recent state-of-the-art methods, going from deterministic parsing to beam search for transition-based parsing and from arc-factored to higher-order models for graph-based parsing. In this context, it is also relevant to mention previous work, such as Hall et al. (2007) and Attardi and Dell'Orletta (2009), which have tried to improve parsing accuracy by switching or combining parsing directions, which implicitly has the effect of changing the position of the root node (if present).

In conclusion, we believe there may be two methodological lessons to learn from our experiments. The first is that, for certain parsing models, the existence and placement of the dummy root node is in fact a parameter worth tuning for best performance. Thus, for the deterministic arc-eager parser, it seems that we can obtain higher parsing accuracy by placing the dummy root node at the end of the sentence (or omitting it completely) instead of placing it at the beginning in the sentence, as is currently the norm in data-driven dependency parsing. The second lesson is that, since the differences observed between different conditions are sometimes at least as large as the differences considered significant when comparing different parsing models, the status of the dummy root node may be an underestimated source of variation and a variable that needs to be controlled for in experimental evaluations. The current practice of consistently placing the root node at the beginning of the sentence is one way of ensuring comparability of results, but given the arbitrariness of this decision together with our experimental results, it may be worth exploring other representations as well.

Acknowledgments

Thanks to Ryan McDonald, Yoav Goldberg and three anonymous reviewers for useful comments, and to Ryan also for help in modifying MSTParser. Miguel Ballesteros is funded by the Spanish Ministry of Education and Science (TIN2009-14659-C03-01 Project).

References

Attardi, Giuseppe and Felice Dell'Orletta. 2009. Reverse revision and linear tree combination for dependency parsing. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL HLT)*, pages 261–264.

Böhmová, Alena, Jan Hajič, Eva Hajičová, and Barbora Hladká. 2003. The Prague Dependency Treebank: A three-level annotation scenario. In Anne Abeillé, editor, *Treebanks: Building and Using Parsed Corpora*. Kluwer, pages 103–127.

Buchholz, Sabine and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL)*, pages 149–164.

de Marneffe, Marie-Catherine, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC)*, pages 449–454.

Hajič, Jan, Barbora Vidova Hladka, Jarmila Panevová, Eva Hajičová, Petr Sgall, and Petr Pajas. 2001. Prague Dependency Treebank 1.0. LDC, 2001T10.

Hall, Johan, Jens Nilsson, Joakim Nivre, Gülsen Eryiğit, Beáta Megyesi, Mattias Nilsson, and Markus Saers. 2007. Single malt or blended? A study in multilingual parser optimization. In *Proceedings of the CoNLL Shared Task of EMNLP-CoNLL 2007*, pages 933–939.

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

McDonald, Ryan. 2006. *Discriminative Learning and Spanning Tree Algorithms for Dependency Parsing*. Ph.D. thesis, University of Pennsylvania.

McDonald, Ryan, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pages 523–530.

Nivre, Joakim. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pages 149–160.

Nivre, Joakim. 2004. Incrementality in deterministic dependency parsing. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together (ACL)*, pages 50–57.

Nivre, Joakim. 2006. *Inductive Dependency Parsing*. Springer.

Nivre, Joakim, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task of EMNLP-CoNLL 2007*, pages 915–932.

Nivre, Joakim, Johan Hall, and Jens Nilsson. 2006. Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC)*, pages 2216–2219.

Nivre, Joakim and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 99–106.