Natural Language Engineering **13** (3): 261–282. © 2007 Cambridge University Press
doi:10.1017/S1351324907004548 First published online 11 June 2007 Printed in the United Kingdom

Some notes on the PARC 700 Dependency Bank

TOMAS BY
Institut für Informatik, Technische Universität München
Boltzmannstraße 3, 85748 Garching bei München, Germany
e-mail: tomas.by@in.tum.de

(*Received 26 March 2006; revised 17 October 2006; accepted 20 February 2007*)

Abstract

The PARC 700 dependency bank is a potentially very useful resource for parser evaluation that has, so to speak, a high barrier to entry, because of tokenisation that is quite different from the source of the data, the Penn Treebank, and because there is no representation of word order, producing an uncertainty factor of some 15%. There is also a small, but perhaps not insignificant, number of errors. When using the dependency bank for evaluation, it seems likely that these things will cause inflated counts for mismatches, so to obtain more accurate measurements, it is desirable to eliminate them. The work reported here consists of an automatic conversion of the dependency bank into a Prolog representation where the word order is explicit, as well as graphical representations of the dependency trees for all 700 sentences, automatically generated from the Prolog data. As a side effect of the transformation, errors were detected and corrected. It is hoped that this work will lead to more widespread use of the PARC 700 dependency bank for parser evaluation.

1 Introduction

The PARC 700 dependency bank[1] (King *et al.* 2003) is a proposed 'gold standard' for parser evaluation, consisting of 700 sentences from the Penn Treebank (Marcus *et al.* 1999), annotated with grammatical information, including predicate–argument structure, and a complete dependency graph for each sentence, plus some semantic features. The method of evaluation for which this data is intended is the one suggested by Carroll *et al.* (1998): converting the parser output to a set of dependency relations between strings representing the base forms of some of the words in the sentence. These relations are then compared with the correct ones in the Dependency Bank. As long as no word occurs more than once per sentence, this system would seem to work as intended, but because of significant differences in tokenisation between the Penn Treebank and the PARC 700, as well as the absence of an explicit representation of word order in the latter, it is not so easy to determine which particular occurrences, out of the multiple ones of the same word in a sentence, are the same in the two corpora. This obviously means that there is a risk of matching against the wrong words when comparing the dependency relations. Evaluations

[1] http://www2.parc.com/istl/groups/nltt/fsbank/

https://doi.org/10.1017/S1351324907004548 Published online by Cambridge University Press

using the Dependency bank are reported by Kaplan *et al.* (2004), Burke *et al.* (2004) and Briscoe and Carroll (2006). Considerable space in these articles[2] is devoted to the problem of converting between the PARC 700 and the various parser output formats, and the difficulties of comparison.[3] These authors mainly concentrate on the semantics of the dependency relations, while this article argues that inconsistency and the lack of word order representation are also part of the problem. As a first step toward determining the contribution of the latter, software has been written to disambiguate the word order of the PARC 700, that is matching up the tokens in the original Penn Treebank data with the tokens in the Dependency Bank. The results of this automatic disambiguation have been stored in a Prolog format that preserves the word order information.

All the 700 sentences in the Dependency Bank are from section 23 of the Penn Treebank, and the identification symbols reflect this. For brevity, only the sequence number is used in this article to identify the sentences.

'parc_23.1' = '#1'
'parc_23.2' = '#2'
'parc_23.3' = '#3'
...

When a reference is made in the dependency bank to the original PTB files, the file name and sequence number of the sentence are given. Using a normal editor, however, it is not necessarily easy to find the right sentence, as the editor would need to parse the Lisp syntax. A more directly useful, and more traditional, approach, adopted here, is to give the file name and line number.[4]

This article consists of two main parts: five sections on more general issues, followed by five sections with more detailed discussion of the various problems with the Dependency Bank. Section 2 discusses the formats used for the representation in the Penn Treebank and PARC 700, and section 3 gives an overview of the problems that the article concentrates on. The concept of 'projectivity' and its use for validating dependency representations are discussed in section 4, followed by a section on the automatic conversion of the PARC 700 into Prolog format, and one section on this format itself. Sections 7–11 present the problems with markup errors, tokenisation inconsistencies, specific lacks of information in the PARC 700, tokenisation differences between it and the Penn Treebank, and finally inconsistencies in the dependency representations. The final section summarises some general conclusions.

[2] Burke *et al.* (2004, pp. 107–15); Kaplan *et al.* (2004, pp. 102–3); Briscoe and Carroll (2006, pp. 4–5).

[3] For example, 'There are a number of reasons for the poorer results against the PARC 700, most of which are related to differences between the representations used in the automatically-generated f-structures and the PARC 700 which could not be captured using the systematic mappings of the conversion software' (Burke *et al.* 2004, p. 116).
'The comparison of systems using DepBank is not straightforward, so we extend and validate DepBank [...]' (Briscoe and Carroll 2006, p. 1).

[4] The line number of the start of the sentence in the 'merged' files (Release 3).

https://doi.org/10.1017/S1351324907004548 Published online by Cambridge University Press

2 Forms of representation

In the type of parser evaluation for which the Penn Treebank was designed (Black *et al.* 1991), it is assumed that the parser uses the same tokenisation as the Treebank, and the evaluation involves comparing the structure in the parser output with that in the Treebank. Various problems with this evaluation method, summarised in Carroll *et al.* (1998), led these authors to suggest an alternative (Carroll *et al.* 1998, 1999), where the data that is compared consists of a set of dependencies between the words in the sentence. This format was then the inspiration for the PARC 700 Dependency Bank (Crouch *et al.* 2002, p. 67). In the representation of Carroll *et al.*, words are represented as strings. As King *et al.* (2003, p. 4) and Crouch *et al.* (2002, p. 71) point out, multiple occurrences of the same word in a sentence can then not be distinguished. This means that if these words have different dependencies, which is likely,[5] the matching might be incorrect. The problem is aggravated by the fact that the words are lemmatised so that even if two occurences of a word have different surface form they will have the exact same representation in the system of Carroll *et al.* To solve this problem, the PARC 700 representation adds a numerical index to each of the words that partake in dependencies. As will be discussed later, problems however remain, because the order of the occurrences is not included.[6]

For technical reasons,[7] a distinction is made in PARC 700 between words that are represented as tokens, and have an index, and words that are represented only by attributes (of the tokens). Determiners and punctuation typically belong to the second class, which seems uncontroversial, but there are other cases, discussed later, where the choice seems arbitrary and unmotivated.

Figure 1 shows the representation of the same sentence (#335) in the two systems, and Table 1 lists various types of information, and their encoding. Apart from the word order, the PARC 700 generally contains slightly more information than the Penn Treebank. The main categories of added data would seem to be the explicit indication of proper names,[8] and the sentence form. Another difference, which is perhaps the most important in practice, is the representation of subordination (and clausal complements) by a simple dependency link in PARC 700. In the PTB these are typically recursive 'S' elements, with empty subjects, and there are many different configurations, so recognising the correct linguistic type from the markup is not always easy. In some cases, information has been lost in the transformation to the PARC 700 format. Sentence #27, shown in Penn Treebank format in Figure 2

[5] In fact, there appears to be only one single case in the PARC 700, where multiple occurrences of the same word have the same relations: 'very' in sentence #583.

[6] 'word order is not indicated other than in the recording of the original string in the *sentence_form* field of each Depbank structure' (King *et al.* 2003, p. 5).

[7] The 'real' tokens are values of the 'pred' attribute in the top level of the LFG f-structures from which the PARC 700 was created, while the 'attribute tokens' correspond to other attribute values in the f-structure (King *et al.* 2003, pp. 2–3).

[8] Proper names are, of course, distinguished from common nouns in the PTB by the part-of-speech code, but there are many, many different structures, and there does not seem to be an easy, reliable way to find just the name (as opposed to, say, a noun phrase containing a noun premodifier that happens to be a name).

https://doi.org/10.1017/S1351324907004548 Published online by Cambridge University Press

```
( (S                                          sentence(
    (NP-SBJ (NNP Bridget) (NNP O'Brian) )       id(wsj_2308.44, parc_23.335)
    (VP (VBD contributed)                     structure(
      (PP-CLR (TO to)                           mood(contribute~0, indicative)
        (NP (DT this) (NN article) )))          obl(contribute~0, to~2)
    (. .) ))                                    stmt_type(contribute~0, declarative)
                                                subj(contribute~0, Bridget O'Brian~1)
                                                tense(contribute~0, past)
                                                vtype(contribute~0, main)
                                                num(Bridget O'Brian~1, sg)
                                                pers(Bridget O'Brian~1, 3)
                                                proper(Bridget O'Brian~1, name)
                                                obj(to~2, article~4)
                                                ptype(to~2, semantic)
                                                deixis(article~4, proximal)
                                                det_form(article~4, this)
                                                det_type(article~4, demon)
                                                num(article~4, sg)
                                                pers(article~4, 3))
                                            )
```

Fig. 1. Penn Treebank versus PARC 700 representation of sentence #335.

```
( (S (PP-TMP (IN In)
       (NP (NP (DT the) (JJ first) (CD six) (NNS months) )
           (PP (IN of) (NP (DT the) (NN year) ))))
     (NP-SBJ (PRP it) )
     (VP (VBD posted)
       (NP (NP (DT a) (JJ net) (NN loss) )
           (PP (IN of) (NP (QP ($ $) (CD 33.1) (CD million) ) (-NONE- *U*) )))) (. .) ))
```

Fig. 2. Penn Treebank, `wsj_2353`:233 (PARC #27).

Table 1. *Comparison of the information content (sentence #335)*

	Encoded in the [...] using [...]	
Information	Penn Treebank	PARC 700
Word order	Preorder traversal	—
Proper name	NP containing NNPs*	Name is one token, with attribute proper = name
Features, name	—	num/2, pers/2 (Bridget O'Brian)
Past tense verb	VBD	vtype = main, tense = past
Subject	SBJ	subj/2
Argument/adjunct†	PP-CLR	—
Oblique argument	—	obl/2
Prepositional object	NP inside PP	obj/2
Determiner	DT	det_form/2, det_type/2
Common noun	NN	num/2, pers/2 (article)
Sentence form	—	stmt_type = declarative, mood = indicative

* There are many variations on this structure in the Penn Treebank.

† Bies (1995, pp. 46–7).

https://doi.org/10.1017/S1351324907004548 Published online by Cambridge University Press

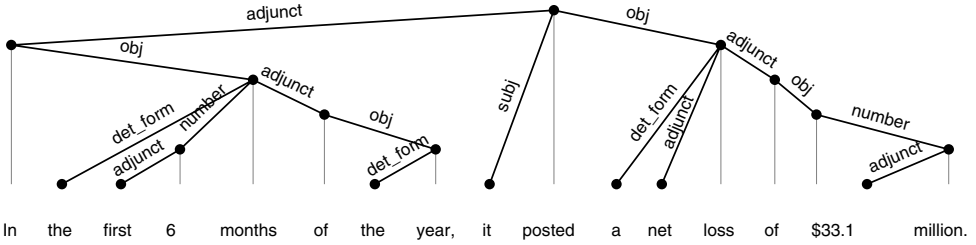


Fig. 3. PARC #27 as a graphical 'tree'.

and in PARC 700 format, as a graphical tree,[9] in Figure 3, has a temporal sentence modifier in the form of a prepositional phrase 'in the first six months of the year'. In the Penn Treebank, this is indicated by the 'TMP' tag (Bies 1995, pp. 44–6), but in the Dependency Bank there is no representation of the fact that the modifier clause is temporal.

The PARC 700 differs from traditional dependency grammar such as Hays (1964, p. 512), Gaifman (1965, p. 306), and Robinson (1970, p. 260) in that it allows 'empty nodes' that do not correspond to any word in the sentence. These are used to represent ellipsed subjects and objects, and, in some cases, coordination, somewhat similar to the conjunction phrase nodes in Hudson (1990, chap. 14).

3 Overview of problems

The problems discussed here have in common that they seem likely to cause spurious mismatches when the PARC 700 Dependency Bank is used for parser evaluation, but they differ in probable cause and severity, which naturally influences the proper response to them. In the case of coding mistakes, of which there is a handful, discussed in section 7, it seems clear that they will cause problems and should be corrected. The most immediately problematic issue, when attempting to use PARC 700 for the first time with a particular parsing system, is probably the tokenisation differences compared to the Penn Treebank (section 10). Only slightly more than a third of the PTB tokens are identical in the PARC 700, and another third have been modified but correspond one-to-one. Those tokens that have disappeared are mainly punctuation characters. It is possible that this is a nonissue if the PARC 700 is used strictly as intended (i.e. as suggested by Carroll *et al.* 1998) but, on the other hand, the alternative Prolog format presented in this paper (section 6) uses the PTB tokens and contains all the PARC 700 information, thus getting the best of both worlds. For what it may be worth, Crouch *et al.* (2002, pp. 72–3) discuss at some length tokenisation problems encountered when using another dependency bank to evaluate the LFG system from which the PARC 700 was created.

The problem that might ultimately turn out to be the most serious is the lack of explicit representation of word order. Because of this, if a word, including

[9] While not strictly a tree in the mathematical sense, but a directed acyclic graph, it has a strong tree-nature, with one root node (cf. Matthews 1981, p. 81)

https://doi.org/10.1017/S1351324907004548 Published online by Cambridge University Press

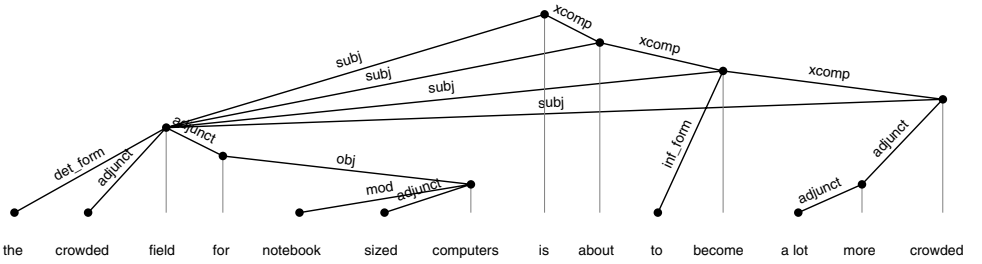


Fig. 4. Graphical 'tree' dependency representation of sentence #284.

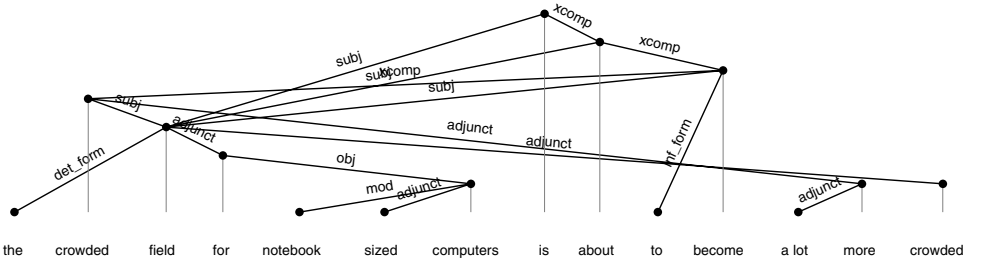


Fig. 5. PARC #284 with the word 'crowded' incorrectly disambiguated.

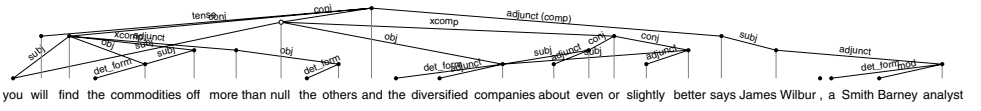


Fig. 6. PARC #622.

morphological forms, occurs multiple times in the same sentence, there is no simple way to decide which is which. The problem is illustrated by Figures 4 and 5, showing two analyses of the same sentence differing only in that the placement of the word 'crowded' has been swapped. Links that are different in the two pictures are those that are affected by the ambiguity. The total number of PARC 700 tokens that are ambiguous in this way is about 15%.

Figure 6 illustrates a couple of other problems. There is a nonleaf empty node (glossed as 'null'), apparently representing the fact that both conjuncts have the same subject. This is problematic because such a node will not necessarily exist in the output of other parsing systems. If that is the case, then presumably none of the four dependency links to and from that node will match correctly, even if there are links, such as a 'subj' and 'obj' to the conjunct, with the same (or very similar) meaning. Finally, the comparative construction 'more than' is a single token in sentence #622 (Figure 6), and in seven other sentences (cf. Table 5), while it is two separate tokens in a further six cases (cf. p. 17ff). There are quite a few examples, discussed in sections 8 and 11, of this type of inconsistency in the PARC 700, but it is not clear if this reflects problems, or if the different forms simply are the best way, in each case, to encode the linguistic structure in dependency format.

https://doi.org/10.1017/S1351324907004548 Published online by Cambridge University Press

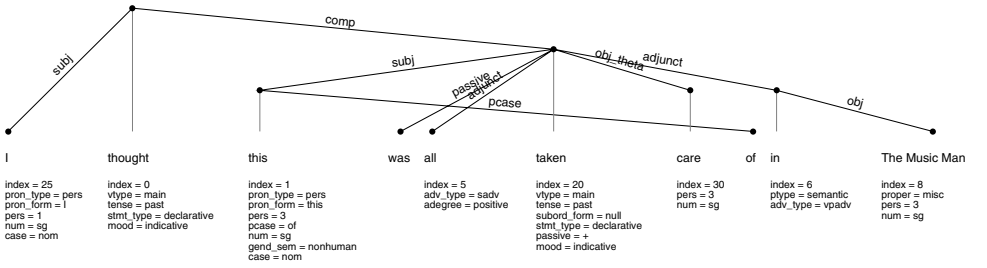


Fig. 7. PARC #113 with misattached attribute.

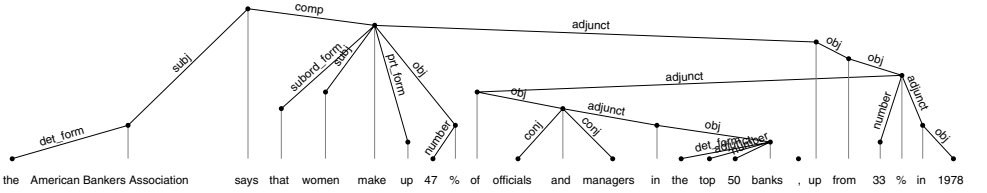


Fig. 8. PARC #104 with incorrect ‘adjunct’ link.

4 Projectivity

The concept of projectivity was apparently discovered around the same time by several groups of researchers, independently,[10] and is generally quite helpful when analysing dependency structures. Mel’čuk and Pertsov (1987, p. 183) offer a succinct formulation. A dependency tree representation of a sentence is projective if

- no two branches of the tree cross one another and
- no vertical projection from a node onto a horizontal line drawn below the tree crosses a branch of the tree.

In sentence #113 (Figure 7), for example, the subtree under the token this/1 is nonprojective, which can easily be seen by the fact that the dependency link from ‘of’ to ‘this’ crosses two other dependency links and two ‘vertical projection lines’. Here, this indicates an error in the markup (cf. section 7), and the link from ‘of’ should go to care/30 instead. Similarly, in sentence #104 (Figure 8) the phrase headed by ‘of’ has been incorrectly attached to the second, instead of the first, occurrence of ‘%’, and the subtree under that token is nonprojective (it crosses two vertical projection lines).

While a number of phenomena in natural language are known to be nonprojective (Mel’čuk and Pertsov 1987, pp. 184–6; Mel’čuk 1988, pp. 36–8), it is almost certain that a sentence with a fully projective dependency tree does not contain any

[10] According to Marcus (1965, p. 181), first published by Harper and Hays in 1959; to Mel’čuk and Pertsov (1987, p. 183) by Hays and by Lecerf, in 1960; and to Dikovsky and Modina (2000, p. 73), by Tseitin *et al.* and by Harper and Hays, in 1959.

https://doi.org/10.1017/S1351324907004548 Published online by Cambridge University Press

errors of the types discussed in this article, such as wrong disambiguations or misattachments. Unfortunately there are several systematic structures in the PARC 700, like the multiple 'subj' links in Figures 4 and 5, that produce nonprojectivities. Even after software-mechanically removing as many of these as seemed achievable with reasonable efforts, the number of nonprojective sentences is still about one hundred, most of which do not contain any obvious problems.

5 Conversion and disambiguation

The conversion from the original PARC 700 format to the Prolog dependency format is done in two steps. First, the original sentence as a string, taken from the 'sentence_form' attribute, is tokenised and matched against the PARC 700 tokens. Because of the (in principle) many-to-many mapping between PTB and PARC 700 tokens, this matching is, in effect, a search procedure that terminates when a complete tokenisation of the string is found that contains all the tokens that occur in the PARC 700 representation of the sentence. The result of this first step is an intermediate representation, with ambiguous tokens, and an example (sentence #3) is shown below. The sentence contains six proper tokens (0, 2, 5, 6, 8, 16) and three attribute tokens (the, the & were), shown with the token index and attribute that they are licensed by

```
the[2:det_form=the] | the[8:det_form=the]
following/16
issues/2
were[0:passive=+]
recently/5
filed/0
with/6
the[2:det_form=the] | the[8:det_form=the]
'Securities and Exchange Commission'/8
```

As can be seen here, the two occurrences of 'the' are ambiguous, since there is no indication of which word, out of the two possibilities (2 & 8), it is that they modify. To resolve this ambiguity, a straight-forward scoring procedure, the second step of the conversion, is used that computes a cost for each ambiguous word, based on the distance, in words, of the dependency links.

In our example, sentence #3, there are two ambiguities, each with one link. From the first 'the' to issues/2 we count two words, and to 'S. & E. C.'/8 eight words. We proceed similarly for the second 'the' and we end up with the following distances:

	issues/2 (word 3)	'S. & E. C.'/8 (word 9)
the (word 1)	2 words	8 words
the (word 8)	-5 words	1 word

Apart from the fact that negative distances are not allowed for determiners (which is to say, those dependency links cannot go to the left), it is clear that the minimum total link length is achieved with the following ordering of the formerly ambiguous

https://doi.org/10.1017/S1351324907004548 Published online by Cambridge University Press

tokens:

```
the[2:det_form=the]
the[8:det_form=the]
```

The disambiguated representation of the sentence is then the following:

```
the[2:det_form=the] following/16 issues/2 were[0:passive=+]
recently/5 filed/0 with/6 the[8:det_form=the]
'Securities and Exchange Commission'/8
```

This is a very simple example, and in most cases there are multiple dependency links to and from the ambiguous words. The average length of the links per word is then used, and there are some extra weights in certain cases. An amount of practical experimentation was involved in the design of this scoring procedure.

The procedure successfully resolves all ambiguities in the PARC 700 dependency bank, with three exceptions. Although there is no guarantee that the resulting disambiguations are correct, the method used, combined with the fact that the tool reports projectivity violations (cf. section 4) gives a certain amount of confidence. In the 'correct' version of sentence #284 (Figure 4), for instance, the only nonprojective structures are the 'extra' subject links. In Figure 5, on the other hand, there are many crossing links, and dependencies from one end of the sentence to the other. Manual inspection of the trees for those sentences that violate projectivity (about 100 out of the 700) is not totally impractical, and a tree such as Figure 5 would stand out.

The three sentences that cannot be automatically disambiguated are #16, #580, and #583. The first has two tokens 'and', one as a sentence-initial adverbial and one as a conjunction. Because the adverbial is only linked to the main verb, which is after the conjunction, the procedure makes the wrong choice. Sentence #583 contains two occurrences of 'very'[11] that modify the same word and thus have the same score. Sentence #580 is hugely ambiguous, and while the processing may eventually succeed, it would take an unreasonable amount of time. By hard-coding the ordering of one of the seven ambiguous token types in that sentence, the rest are disambiguated quickly.

6 Prolog representation

While writing code to parse the PARC 700 Dependency Bank files is not particularly difficult, it seems unnecessary to distribute data in a syntax for which no tools are available. If it is distributed in a standard programming language syntax such as Common Lisp or Prolog, then any problems with the markup (cf. Table 2) will be reported as errors by the interpreter, and it is a simple matter to write code to print the data in whatever format is required.

Because a tool is available to create graphical dependency 'trees' from the Prolog representation in Figure 9, this has been used for the conversion of PARC 700.

[11] As mentioned above (p. 263).

https://doi.org/10.1017/S1351324907004548 Published online by Cambridge University Press

Clause ::= `sentence(` *SentNum*, *IdString*, *WordNums*, *NodeNums* `)`
| `word(` *SentNum*, *WordNum*, *Word*, *Attributes* `)`
| `node(` *SentNum*, *NodeNum*, *Word*, *Attributes* `)`
| `dependency(` *SentNum*, *Depnode*, *Dependency*, $Depnode_{to}$ `)`
SentNum ::= Number (Identifying the sentence)
WordNum ::= Number (Identifying the word)
NodeNum ::= Number (Identifying the empty node)
IdString ::= Atom (Arbitrary string identifying the sentence)
WordNums ::= List of numbers (In the right sentence order)
NodeNums ::= List of numbers (In arbitrary order)
Word ::= Atom
Attributes ::= List of pairs of atoms
Depnode ::= `w(` *WordNum* `)` | `n(` *NodeNum* `)`
Dependency ::= Atom (Name of the grammatical relation)

Fig. 9. The format of the Prolog dependency representation.

Other than correction of the errors mentioned in this article, no modifications have been made of the attributes or dependencies. The only additional information is the word order. The original PARC 700 indexes, which have been separated from the token strings, are added as attributes instead. Here follow some general principles of the conversion.

- Unlike in the PARC 700, all the characters in the Penn Treebank sentences are included in the tokens, but there is a many-to-many relation between these and the original Penn Treebank tokens.
- Tokens are numbered (consecutively, but the word order is also encoded by the list in the `sentence`/4-clause). The token numbers (or 'word numbers') are not the same as the PARC 700 indexes.
- Since the PARC 700 tokens do not always include all characters in the original sentence, for some sentences, some tokens in the Prolog representation will be unconnected.

Figure 10 shows the PARC 700 and Prolog representations of sentence #34, and Figure 11 is the graphical 'tree'.

The converted and slightly corrected PARC 700 dependency bank, in Prolog format, can be downloaded from the Web page:

`http://www.basun.net/homepages/tomas/papers/parc700/`

There are two files: '`parc700.pl`' is the Prolog data, in the format shown in Figure 9, and '`parc700.pdf`' contains the graphical 'trees' of all the 700 sentences in the PARC 700.

Because of tokenisation differences, there might still be a need for conversion or mapping to use the data for parser evaluation, but in any case it should be easier to start from the Prolog version than from the PARC 700 format. Since the Prolog tokens contain exactly the same sequence of characters as the corresponding Penn Treebank sentences, the mapping of the tokens can be done relatively easily by looping from left to right. There is no need for any disambiguation procedure,

https://doi.org/10.1017/S1351324907004548 Published online by Cambridge University Press

```
sentence(
  id(wsj_2356.19, parc_23.34)
  date(2002.6.12)
  validators(T.H. King, J.-P. Marcotte)
  sentence_form(The device was replaced.)
  structure(
    mood(replace~0, indicative)
    passive(replace~0, +)
    stmt_type(replace~0, declarative)
    subj(replace~0, device~1)
    tense(replace~0, past)
    vtype(replace~0, main)
    det_form(device~1, the)
    det_type(device~1, def)
    num(device~1, sg)
    pers(device~1, 3)
  )
)
```

```
sentence(34,'PARC#34 (wsj_2356:422)',
         [0,1,2,3],[]).
word(34,0,the,[]).
word(34,1,device,[index-'1',
                  pers-'3',num-sg,
                  det_type-def,
                  det_form-the]).
word(34,2,was,[]).
word(34,3,replaced,[index-'0',
                    vtype-main,
                    tense-past,
                    stmt_type-declarative,
                    passive-'+',
                    mood-indicative]).
dependency(34,w(0),[det_form],w(1)).
dependency(34,w(1),[subj],w(3)).
dependency(34,w(2),[passive],w(3)).
```

Fig. 10. PARC 700 versus Prolog representation of sentence #34.

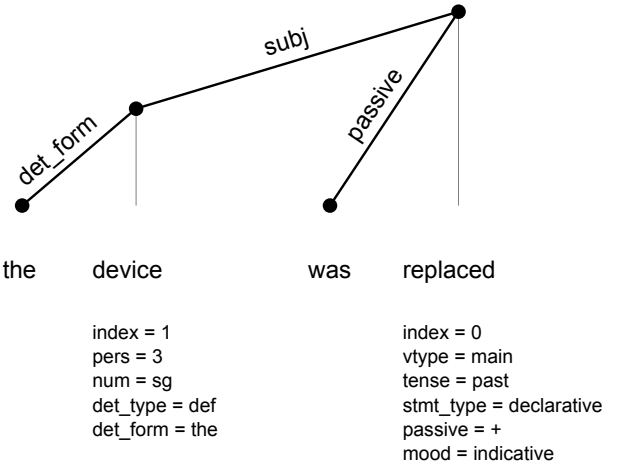


Fig. 11. Graphical tree representation of PARC#34.

as there is if starting from the PARC 700 files. In the latter there is also no clear distinction between real tokens and empty nodes, so that is another disambiguation step that is not necessary when using the Prolog representation. Finally, the attributes in the Prolog data are stored in the same clause as the word itself. In the PARC 700 files they are spread out individually, with a format that is quite similar to the dependencies, so collecting them is a third processing component that can be avoided by using the Prolog format.

7 Errors/mistakes

Table 2 lists the markup errors that were found in the PARC 700 files. In addition to the dependencies between tokens, the PARC 700 contains a large number of attributes of single tokens. As a side effect of the conversion, some errors in these were also discovered. When collecting the words that are encoded as attributes, as

https://doi.org/10.1017/S1351324907004548 Published online by Cambridge University Press

Table 2. *Markup errors in the dependency bank*

Sentence(s)	Problem
#44	Of the seven occurrences of the word with index '9', in four the token is 'fall' and in three it is 'fell'
(Seven)	Attribute values ('how many'/'how far'/'no one'/'why not') contain spaces, which, according to the documentation, is not allowed
#262,#263	The word 'and' occurs in a token that contains whitespace, but with the wrong capitalisation
#304	Missing comma in the 'validators' field
#410	'Vagabonds Hotels' tokenised as 'Vagabond Hotels'
#539	Missing comma in the 'id' field
#550, #627, #684	The 'sentence_form' field contains only the text up to the first right parenthesis in the sentence

Table 3. *Representation errors in the PARC 700 dependency bank*

Sentence(s)	Problem
#65	No token or attribute for the determiner 'a'
#104	The dependency link from of/7 should go to %/6, not %/34.
#113	The attribute 'pcase=of' should be on care/30, not this/1.
#116	No token or attribute for the determiner 'the'
#198, #535	Duplicate attribute 'proper' (misc/date)
#243, #488	No token or attribute for the preposition/adverb 'as'
#249	Duplicate attribute 'subord_form'
#272	Extra attribute 'subord_form=that' (The word 'that' does not occur in the sentence)
#464	No token or attribute for the conjunction 'that'
#550	Extra token 'well' (the word occurs only in the phrase 'as well as' which is also a token)
#578	The Penn Treebank token '1/4' has become '0 1/4'
#590	Extra token '59' (it occurs only in '11:59' which is also a token)
#603	An 'it' has been added which is not in the original sentence
#634	The word 'lose' in 'having lost' is marked as progressive

opposed to indexed tokens, any attributes that have no corresponding token are indicated. Conversely, words in the sentence that are neither PARC 700 tokens nor attribute tokens are also reported. Table 3 lists the problems found in this manner and a couple of additional oddities that have been noted.

Progressive aspect is, according to the PARC 700 manual, encoded by an attribute 'prog=+' and the auxiliary verb (be) is not a token. Examination of the data seems to indicate that this attribute occurs with any present participle, including noun phrase modifiers, regardless of whether or not there is an auxiliary verb. Similarly, the attribute 'passive=+' seems to occur with any past participle and not just those that are preceded by 'be'. For progressive aspect, the manual clearly states that the attribute implies that there is a form of 'be' in front of the verb, and there are 166 instances when this is not the case. These are not listed in Table 3, but they have been removed from the data as part of the conversion process. The manual says of the 'passive=+' attribute only that it indicates 'passive verbs', not that it implies the existence of an auxiliary, so this attribute has not been altered.

Other problems that were discovered because of projectivity violations, besides Figures 7 and 8, include the temporal adverbial 'Friday' in sentence #470 (Figure 12)

https://doi.org/10.1017/S1351324907004548 Published online by Cambridge University Press

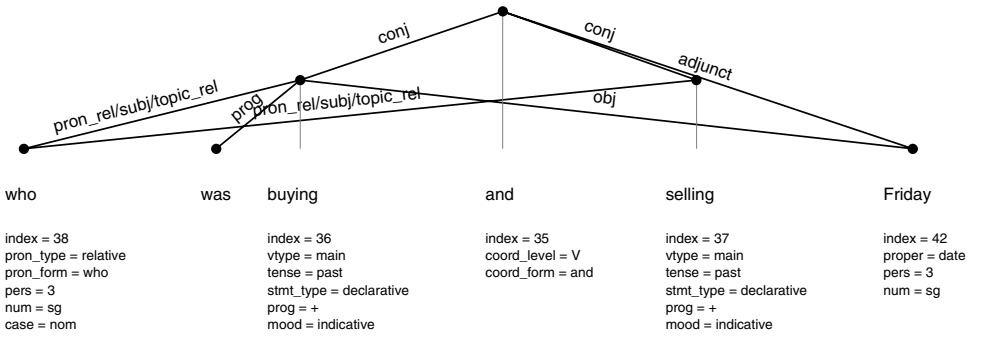


Fig. 12. Fragment of sentence #470 with conjunction and adverbial.

which has been encoded as both the object (obj) of one of the verbs in the conjunction (but not the other) and the 'adjunct' of the conjunct itself. It would seem that it has to be one or the other, and that there should be another 'obj' link to the second verb in the former case. A third possibility would be to make it an 'adjunct' of both verbs.

8 Inconsistencies

Table 4 lists some examples of hyphenated tokens in the PARC 700, revealing that their analysis is not totally consistent. There are other inconsistencies, and some examples are shown here. In sentence #323 (left, below) the adverbial phrase 'right now' is one token, but in sentence #116 (and #183) it is two tokens.

```
adjunct(be~2, right now~27)
adegree(right now~27, positive)
adv_type(right now~27, sadv)
```

```
adjunct(be~0, now~8)
adjunct(now~8, right~7)
adegree(right~7, positive)
adegree(now~8, positive)
adv_type(right~7, advmod)
adv_type(now~8, sadv)
```

Likewise, the phrase 'how far' is one token in sentence #330 (left) and two in sentence #419 (right).

```
adv_type(how far~36, amod-int)
pron_form(how far~36, how far)
pron_type(how far~36, interrogative)
```

```
adjunct(far~12, how~16)
adv_type(how~16, amod-int)
pron_form(how~16, how)
pron_type(how~16, interrogative)
```

Table 4. *Examples of inconsistent hyphenation*

Penn Treebank token	Unchanged in	Dehyphenated in
big-selling	#681	—
biggest-selling	—	#461
deficit-reduction	#127, #594, #596	#128
desk-top	#541	#542
long-term	#54, #563, #582	#584
principal-only	#626, #628	#338
two-year	—	#593, #625
four-year	—	#676
10-year	—	#625
30-year	#227	—

https://doi.org/10.1017/S1351324907004548 Published online by Cambridge University Press

Table 5. *Tokens containing spaces that are not proper names*

Token	Occurs in	(But not in)
'a few'	#387	
'a little'	#150	
'a lot'	#284	
'a mere'	#603	#636
'as many as'	#673	
'as much as'	#21, #349, #609	
'as soon as'	#341	
'as though'	#685	
'as well as'	#425, #516, #536, #550, #575, #577	
'at least'	#472, #506, #507	#657
'better than'	#178	
'even more'	#421	
'even though'	#593	#527, #691
'flat out'	#359	
'for example'	#118	
'for instance'	#63	#56, #88, #106, #114, #549, #577
'how far'	#330	#419
'how many'	#273	
'less than'	#603	#78
'many more'	#298	#273
'more than'	#366, #514, #526, #572, #593, #622, #627, #649	#357
'no one'	#88, #205, #251, #353	
'of course'	#298, #315	
'once again'	#510	#10
'rather than'	#99, #102, #181, #202, #453, #487, #568, #632	
'right now'	#323	#116, #183
'sell orders'	#423, #576	
'so much'	#326	
'such as'	#99, #138, #250, #365, #370, #556	
'that is'	#300	
'the most'	#531	#62, #116
'up to'	#5	
'why not'	#186	

Several similar cases are listed in Table 5. From the available PARC 700 documentation it is hard to tell whether or not this apparent inconsistency is intended, but in general it would seem preferable to give each linguistic construction one consistent interpretation. On the other hand there was presumably a reason in each case for doing it the way it was done.

Yet another type of construction in the PARC 700 that is nonprojective is the one exemplified by sentence #169 (Figure 13),[12] where the words 'more' and 'than' are linked in the dependency 'tree' but not contiguous in the sentence. There are also several cases[13] in which those words, in similar contexts, are not linked directly. In

[12] And also by sentence #300. There are similar, equally nonprojective, structures in #402 (milder than), #521 (steeper than) and #650 (less robust than).
[13] In sentences #493 (Figure 14), #521 and #598.

https://doi.org/10.1017/S1351324907004548 Published online by Cambridge University Press

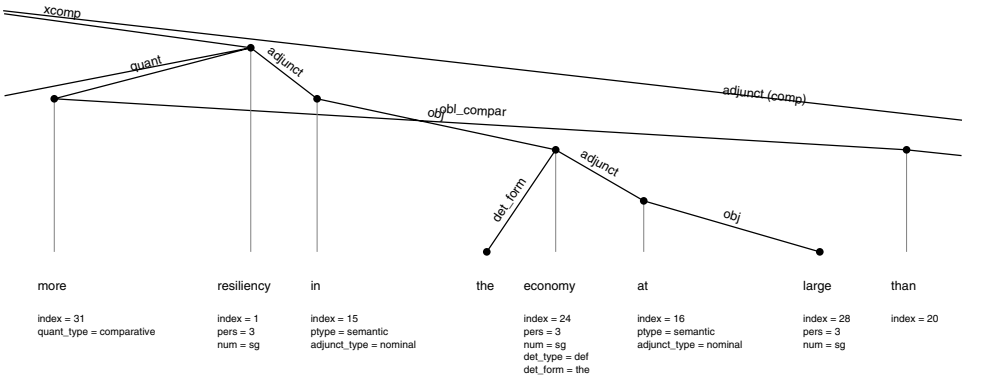


Fig. 13. Fragment of sentence #169 with nonprojective 'more-than' link.

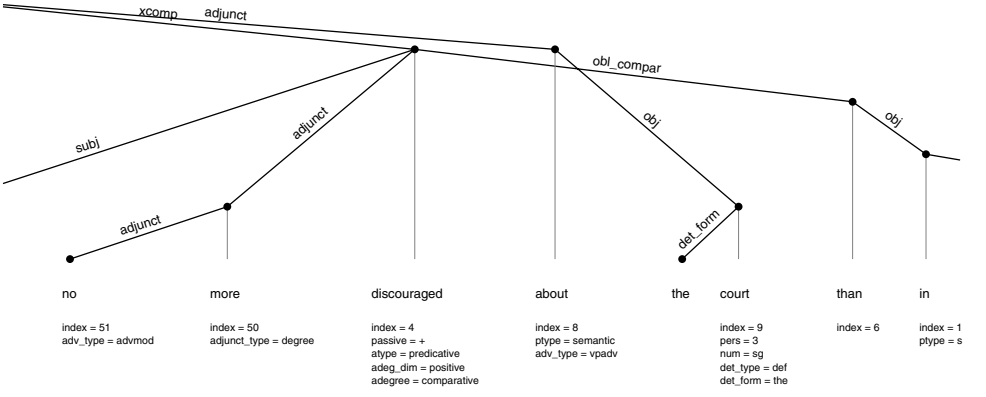


Fig. 14. Fragment of sentence #493, with unconnected 'more-than' tokens.

addition, there is one sentence (#357) in which the (linked) words are next to each other, and several in which the two words are one token.[14]

9 Shortcomings

As mentioned earlier, and as illustrated by Figures 4 and 5, the PARC 700 dependency bank lacks an explicit representation of word order. How important this is in practice, for parser evaluation, is an empirical question. To take an obvious example, few parsers are likely to mis-attach two determiners (which are words that commonly occur multiple times) to each other's nouns. But the first question must be how many times the same word, including morphological variants, occurs more than once in the same sentence, and those figures are available. There are 112 occasions when a word occurs exactly twice in one single sentence only, and Table 6 lists the tokens that are more ambiguous than that. In total, there are 2,425 tokens, including attributes, which is 18%, whose position in the sentence cannot

[14] Cf. Table 5, and Figure 6.

https://doi.org/10.1017/S1351324907004548 Published online by Cambridge University Press

Table 6. *Tokens that occur multiple times in the same sentence*

Token	Number of occurrences (in one sentence)							
	Twice	3 times	4 times	5 times	6 times	7 times	8 times	12 times
'$'	22	4	1	1	—	—	—	—
'%'	16	1	—	1	—	—	—	1
','	140	51	17	9	2	2	1	—
'.'	1	—	2	1	—	—	—	—
'8 11/16'	—	—	1	—	—	—	—	—
'8 13/16'	—	1	—	—	—	—	—	—
'U.S.'	3	—	—	—	—	—	—	—
a	51	15	2	—	—	—	—	—
about	2	—	—	—	—	—	—	—
and	30	3	2	—	—	—	—	—
as	4	—	—	—	—	—	—	—
at	4	—	—	—	—	—	—	—
billion	4	—	—	—	—	—	—	—
by	4	1	—	—	—	—	—	—
can	2	—	—	—	—	—	—	—
cents	5	1	—	—	—	—	—	—
for	9	1	—	—	—	—	—	—
from	2	—	—	—	—	—	—	—
futures	2	—	—	—	—	—	—	—
he	3	—	—	—	—	—	—	—
her	2	1	—	—	—	—	—	—
in	38	7	—	—	—	—	—	—
is	10	1	—	—	—	—	—	—
it	13	1	—	—	—	—	—	—
its	5	—	—	—	—	—	—	—
million	12	—	—	—	—	—	—	—
months	—	—	—	1	—	—	—	—
net	2	—	—	—	—	—	—	—
not	2	—	—	—	—	—	—	—
of	74	12	2	—	—	—	—	—
on	5	—	—	—	—	—	—	—
or	6	—	—	—	—	—	—	—
our	2	—	—	—	—	—	—	—
price	2	—	—	—	—	—	—	—
prices	3	—	—	—	—	—	—	—
programs	—	1	—	—	—	—	—	—
revenue	2	—	—	—	—	—	—	—
share	5	—	—	—	—	—	—	—
shares	3	1	—	—	—	—	—	—
she	5	—	—	—	—	—	—	—
steel	2	—	—	—	—	—	—	—
stocks	3	—	—	—	—	—	—	—
tax	—	1	—	—	—	—	—	—
than	2	—	—	—	—	—	—	—
that	16	2	—	—	—	—	—	—
the	129	49	17	6	3	—	—	—
they	6	—	—	—	—	—	—	—
this	2	—	—	—	—	—	—	—
to	49	10	1	—	2	—	—	—
up	2	—	—	—	—	—	—	—
was	8	—	—	—	—	—	—	—
we	2	—	—	—	—	—	—	—
were	2	—	—	—	—	—	—	—
will	5	—	—	—	—	—	—	—
with	4	—	—	—	—	—	—	—
would	2	1	—	—	—	—	—	—
year	2	—	—	—	—	—	—	—
you	3	—	—	—	—	—	—	—

https://doi.org/10.1017/S1351324907004548 Published online by Cambridge University Press

Table 7. *Tokenisation of the 700 sentences*

	Words/phrases		Punctuation*		Empty	
	Tokens†	Attributes‡	Tokens†	Attributes‡	nodes§	Total
Penn Treebank	13915	—	2215	—	—	16130
PARC 700	11039	2026	213	7		13285
(PARC 700)					407‖	

* Tokens that contain only punctuation characters.
† Words or phrases that have an index in the PARC 700 Dependency Bank.
‡ Words or phrases that do not have an index but are attributes of other tokens.
§ Indexed tokens that are not words in the sentence.
‖ This includes the presumably erroneous 'rate' in #333 (cf. note 19 on p. 18).

be determined by comparing with the string. Ignoring tokens that contain only punctuation characters, it is still about 15%. It is not clear, however, what impact this can be expected to have on the measurements of parser accuracy.

10 Tokenisation differences

In contrast to the Penn Treebank, where the tokens are substrings of the original text, the tokens in the PARC 700 are in many cases modified, typically by being converted to the base form.[15] There are also significant differences in hyphenation and the role of whitespace. As a rule, the latter never occurs in the Penn Treebank tokens, but in the PARC 700 it does, mostly in proper names. Finally, quite a few PTB tokens, generally punctuation, do not occur in the corresponding PARC 700 sentences at all. Table 7 summarises the differences.

In many cases, a single token in the PTB corresponds to two or more tokens in the PARC 700, as in example one below. There are also a number of cases in which the reverse is true, as in example two below. Most proper names are of this type. As examples three and four below show, there are some extreme cases of nondirect token correspondence.

1. Penn Treebank: '34-year-old' (wsj_2386: 518)
   PARC 700: '34' + 'year' + 'old' (#269)
2. Penn Treebank: '34th' + 'Street' (wsj_2346: 622)
   PARC 700: '34th Street' (#484)
3. Penn Treebank: 'New' + 'York-based' (wsj_2339: 296, wsj_2381: 602)
   PARC 700: 'New York' + 'based' (#452, #585)
4. Penn Treebank: '1940-1987' (wsj_2378: 146)
   PARC 700: '1940' + 'to' + '1987' (#566)

There are 69 tokens in the PTB, all involving hyphens, that correspond to more than one token in PARC 700, or 142 tokens altogether (cf. Table 8). Similarly,

[15] In some cases, such as 'assort' in sentence #61, from the adjective 'assorted', this process has been applied a little too overenthusiastically.

https://doi.org/10.1017/S1351324907004548 Published online by Cambridge University Press

Table 8. *Comparison of the tokenisations*

	PTB	PARC	(Both)
Identical			6,571
Same, except for capitalisation			306
Reformatted numbers (e.g. `1.10` → `1.1`)			26
Expanded abbreviations			20
Morphology (e.g. `is` → `be`)			2,369
Other modification (e.g. `n't` → `not`)			105
Double token, modified (e.g. `ca n't` → `can not`)			17 (× 2)
Symbolic tokens (`coord`, `percent`, `pro`)*			1,047†
Attribute tokens			2,007‡
Tokens that correspond directly	12,485	12,485	12,485
Multiword tokens	69	658	
Components of multiword tokens	1,587	142	
Tokens with no correspondence	1,989	1	
Total number of tokens	16,130	13,286§	

* These are listed, together with the corresponding PTB tokens, in Table 10.
† An additional eleven symbolic tokens are part of multiword tokens.
‡ Another twenty-five attribute tokens are among the 'modified' tokens, and one is part of a multiword token.
§ It is unclear why this is one more than in Table 7.

658 of the PARC tokens, mostly proper names,[16] correspond to 1,587 PTB tokens.

The third column of Table 9 gives a breakdown of the 1989 Penn Treebank tokens (cf. Table 8) that have no corresponding token in the Dependency bank. For some of the token types, a few – or even most – of the occurrences are included in PARC 700, while the others are not. In most cases, the included ones are parts of named entity tokens or numbers. The colons and semicolons, and some of the commas, are linked to coordination nodes (coord), as is the missing 'that' in sentence #270.[17] Pronouns and (some) conjunctions have been replaced by symbolic tokens, listed in Table 10, with attributes (`pron_form`/`coord_form`) that contain the actual word. The use of a symbolic token like 'pro' would seem to presuppose that the string in question does not also occur as a real word. But it does, in fact, occur once: in sentence #120, 'a pro like Jackie Mason'. There are also a number of cases,[18] all of them empty nodes, where the '`pron_form`' attribute is missing.

11 Inconsistent dependencies

Empty nodes in the PARC 700 fall into two broad categories. First there are those that represent ellipsed verbs,[19] plus a number of instances,[20] including the fragment shown in Figure 15, of (it would appear) the situation mentioned in the

[16] Table 5 contains those that are phrases consisting of normal words.
[17] Sentence #270 has two occurrences of the word 'that' and neither is a token. Their only representation is an attribute '`subord_form`' on a 'coord' token.
[18] In sentences #6, #99, #129, #295, #311, #321, #387, #520 and #548.
[19] 'null' in #168, #283, #622 (fig. 6); 'rate' in #333. The latter is presumably a mistake.
[20] 'null' in #205; 'null-be' in #121, #140, #184, #403, #466, #515, #531, #662 and #696.

https://doi.org/10.1017/S1351324907004548 Published online by Cambridge University Press

Table 9. *PTB tokens that have no correspondence in the PARC 700*

Token	Occurs in the PTB	Missing in PARC	Occurrences in PARC tokens‖	(coord's)¶
'a'	Many times	#65	Many	
'the'	Many times	#116 × 2	Many	
'that'	Many times	#270, #464	Many	1 (#270)
'what'	Many times	#456	Many	
'as'	Many times	#243, #488	Many	
'and'	Many times	#109, #130, #388	Many	
'do'#	Many times	(20)	Many	
''s'	146 times	(16)	Many (110)	
','	879 times	(867)	Some (30)	8
'.'	679 times	(660)	Many (349)	
';'	22 times	(seven)	0	15
':'	18 times	(15)	2**	3
'?'	Four times	(four)	0	
'!'	Once	(one)	0	
'"'	Eleven times	(one)	Many (122)	
'`'	Once	(one)	0	
'"'/'`''	154 times each	(154 × 2)	0	
'['/'[']	Ten times each	(nine × 2)	1 + 1††	
'{'/'{'}	Four times each	(four × 2)	0	
'--'	45 times	(45)	0	
'...'	Five times	(five)	0	

‖ Including cases when the PTB token is part of a PARC 700 multiword token.
¶ These are attribute tokens linked to 'coord' tokens.
# Includes 'Do', 'did' and 'does'.
** In these two cases (sentences #590 and #643) the colon is part of the Penn Treebank token also.
†† These have become parentheses in the dependency bank (sentence #218).

Table 10. *Symbolic tokens in the PARC 700 Dependency Bank*

Token	Count	Corresponding tokens in the PTB
'pro'	656	Pronouns
'pro' (empty node)	358	—
'coord'	321	'and', 'or', 'nor', 'either', 'neither', 'but', 'both', 'plus', 'as well as', 'v.', ',', ';', ':'
'percent'	80	'%'
'null' (empty node)	39	—
'null-be' (empty node)	9	—

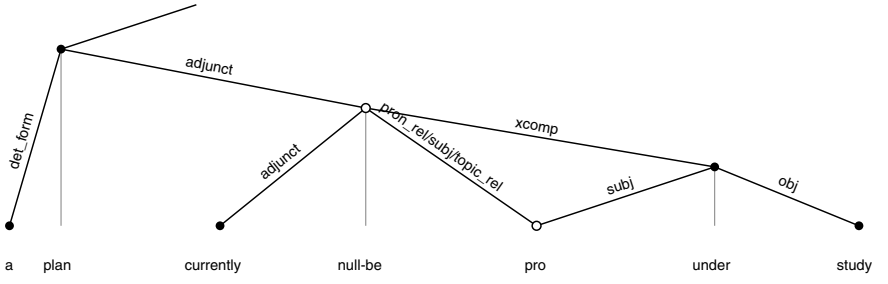


Fig. 15. Fragment of sentence #515 with a nonleaf empty node.

https://doi.org/10.1017/S1351324907004548 Published online by Cambridge University Press

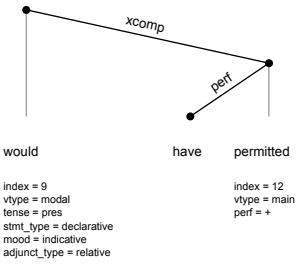


Fig. 16. Fragment of PARC#134.

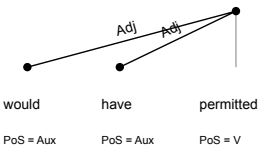


Fig. 17. Rambow *et al.* (2002).

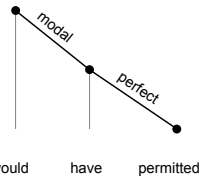


Fig. 18. By.

manual where copular constructions get an extra 'xcomp'. These empty nodes are all nonleaves, which means that they may well create problems when the PARC 700 is used to evaluate parsers that do not also create empty nodes in the same contexts.

The second broad category is the 'pro' empty nodes[21] and the rest of the 'null' nodes. These are leaves in the dependency 'tree' and can therefore more easily be ignored if they are not needed.

Except for 'will', modal verbs in the PARC 700 are treated as the heads of their subtrees, and all other auxiliaries are dependents of the main verb, as shown in Figure 16. This means that if the finite verb is a form of 'be' or 'have', or if it is 'will', the head of the subphrase is the main verb. Other methods of dependency-based modelling of English verb groups have been suggested by Rambow *et al.* (2002, Figure 17) and by the present author (By 2004, Figure 18). All these may have their merits, and it is always possible to convert between them, with varying degrees of effort, but obviously the differences will have to be taken into account when using the data for evaluation. Both Kaplan *et al.* (2004, p. 102) and Burke *et al.* (2004, pp. 104, 113–14) report that they convert the verb group representations.

12 General conclusions

When evaluating a parser by comparing its output to the dependency bank, there must be some way of mapping between the tokens recognised by the parser and those used in the PARC 700. A combination of lemmatising, string-comparison, and a lack of word-order representation is, in effect, a one-to-many mapping from the parser token to all tokens in the PARC 700 sentence that have the same base form, and that means 15% of the word tokens in the PARC 700 are ambiguous. To avoid this uncertainty factor, the corpus should include an explicit representation of word order.

To verify the data, it is generally more productive to rely on automatic tools, as far as possible, rather than on manually inspecting text files in the editor. If Lisp or Prolog syntax had been used for the PARC 700, then simply attempting to read the data into the interpreter would have uncovered the markup errors (Table 2). For a dependency representation, a tool that checks whether or not the 'trees' are projective is useful since 'semantic' problems such as misattachments will typically

[21] Such as the leaf empty node in Figure 15.

https://doi.org/10.1017/S1351324907004548 Published online by Cambridge University Press

result in nonprojective constructions. Even though many, or most, of the projectivity violations in the corpus may not be erroneous, it is quite possible that the remaining ones indicate all the errors.

While automatic projectivity verification is possible only for dependency banks, the problems discussed in this article are not unique to that type of corpora. There are quite a few similar problems in the Penn Treebank, for example, including tagging errors,[22] erroneous subtree labels (syntactic tags), and inconsistencies in the representation.[23] There has not been much discussion of these issues in the extensive literature on PTB-based parser evaluation, presumably because of the general prevalence of auto-generated grammars, where the 'scores' are unaffected since the same errors occur in both the grammar and the corpus. The long-term goal of the enterprise, however, is to parse language correctly, and to that end it must be worthwhile to find and eliminate these problems. It is hoped that the present work will make the PARC 700 dependency bank more accessible, and perhaps offer some suggestions for future producers of dependency-based corpora.

Acknowledgements

Several anonymuos reviewers gave helpful comments, as did Mary Ellen Foster, who also helped with the French sources. Ted Briscoe and Tracy Holloway King commented on an earlier version of the paper.

References

Bies, A. 1995. Bracketing guidelines for Treebank II Style Penn Treebank Project. Technical report, University of Pennsylvania.

Black, E., Abney, S., Flickenger, S., Gdaniec, C., Grishman, C., Harrison, P., Hindle, D., Ingria, R., Jelinek, F., Klavans, J., Liberman, M., Marcus, M., Roukos, S., Santorini, B. and Strzalkowski, T. 1991. Procedure for quantitatively comparing the syntactic coverage of English grammars. In E. Black (ed. *HLT '91: Proceedings of the Workshop on Speech and Natural Language*, pp. 306–11. Morristown, NJ: Association for Computational Linguistics.

Briscoe, T. and Carroll, J. 2006. Evaluating the speed and accuracy of an unlexicalized statistical parser on the PARC Depbank. In *Proceedings of COLING/ACL*, Sydney, Australia.

Burke, M., Cahill, A., O'Donovan, R., J. Genabith, van and Way, A. 2004. The evaluation of an automatic annotation algorithm against the PARC 700 dependency bank. In *Proceedings of the Ninth International Conference on LFG*, Christchurch, New Zealand.

By, T. 2004. English dependency grammar. In *Proceedings of the Workshop on Recent Advances in Dependency Grammar, COLING-2004*, Geneva.

Carroll, J., Briscoe, T. and Sanfilippo, A. 1998. Parser evaluation: A survey and a new proposal. In *Proceedings of the first International Conference on Language Resources and Evaluation.*

Carroll, J., Minnen, G. and Briscoe, T. 1999. Corpus annotation for parser evaluation. In *Proceedings of the EACL-99 Post-Conference Workshop on Linguistically Interpreted Corpora*, Bergen, Norway.

[22] Up to 6% (Taylor, Marcus, and Santorini 2003, p. 17).
[23] Hockenmaier and Steedman (2005, pp. 27, 95–117).

https://doi.org/10.1017/S1351324907004548 Published online by Cambridge University Press

Crouch, R., Kaplan, R., King, T. H. and Riezler, S. 2002. A comparison of evaluation metrics for a broad coverage parser. In *Workshop on Beyond PARSEVAL at the Language Resources and Evaluation Conference*, pp. 67–74, Canary Islands, Spain.

Dikovsky, A. and Modinan, L. 2000. Dependencies on the other side of the curtain. *Traitement Automatique des Langues* **40**(1): 67–96.

Gaifman, H. 1965. Dependency systems and phrase–structure systems. *Information and Control* **8**: 304–37.

Hays, D. G. 1964. Dependency theory: A formalism and some observations. *Language* **40**(4): 511–25.

Hockenmaier, J. and Steedman, M. 2005. CCGbank user's manual. Technical Report MS-CIS-05-09, Department of Computer and Information Science, University of Pennsylvania, Philadelphia.

Hudson, R. 1990. *English Word Grammar*. Oxford, UK: Basil Blackwell Ltd.

Kaplan, R. M., Riezler, S., King, T. H., Maxwell, J. T., Vasserman, A. and Crouch, R. 2004. Speed and accuracy in shallow and deep stochastic parsing. In *Proceedings of HLTC/NAACL*, pp. 97–104.

King, T. H., Crouch, R., Riezler, S., Dalrymple, M. and Kaplan, R. M. 2003. The PARC 700 dependency bank. In *Proceedings of the Fourh International Workshop on Linguistically Interpreted Corpora*, Budapest, Hungary.

Marcus, M. P., Santorini, B., Marcinkiewicz, M. A. and Taylor, A. 1999. Treebank-3 CD-ROM. LDC Catalog No. LDC99T42.

Marcus, S. 1965. Sur la notion de projectivité. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik* **11**: 181–92.

Matthews, P. 1981. *Syntax*. Cambridge University Press.

Mel'čuk, I. A. 1988. *Dependency Syntax: Theory and Practice*. State University of New York Press.

Mel'čuk, I. A. and Pertsov, N. V. 1987. *Surface Syntax of English*. John Benjamins.

Rambow, O., Cresswell, C., Szekely, R., Tarber, H. and Walker, M. 2002. A dependency treebank for English. In *Third International Conference on Language Resources and Evaluation*, pp. 857–63. Las Palmas, Canary Islands.

Robinson, J. J. 1970. Dependency structures and transformational rules. *Language* **46**(2): 259–285.

Taylor, A., Marcus, M. and Santorini, B. 2003. The Penn Treebank: An overview. In A. Abeillé (ed.) *Treebanks: building and using parsed corpora*, pp. 5–22. Dordrecht: Kluwer.

https://doi.org/10.1017/S1351324907004548 Published online by Cambridge University Press