Chapter 9
Large Scale Syntactic Annotation of Written Dutch: Lassy

Gertjan van Noord, Gosse Bouma, Frank Van Eynde, Daniël de Kok, Jelmer van der Linde, Ineke Schuurman, Erik Tjong Kim Sang, and Vincent Vandeghinste

9.1 Introduction

The construction of a 500-million-word reference corpus of written Dutch has been identified as one of the priorities in the STEVIN programme. The focus is on written language in order to complement the Spoken Dutch Corpus (CGN) [13], completed in 2003. In D-COI (a pilot project funded by STEVIN), a 50-million-word pilot corpus has been compiled, parts of which were enriched with verified syntactic annotations. In particular, syntactic annotation of a sub-corpus of 200,000 words has been completed. Further details of the D-COI project can be found in Chap. 13, p. 219. In Lassy, the sub-corpus with verified syntactic annotations has been extended to one million words. We refer to this sub-corpus as Lassy Small. In addition, a much larger corpus has been annotated with syntactic annotations automatically. This larger corpus is called Lassy Large. Lassy Small contains corpora compiled in STEVIN D-COI, some corpora from STEVIN DPC (cf. Chap. 11, p. 185), and some excerpts from the Dutch Wikipedia. Lassy Large includes the corpora compiled in the STEVIN SONAR project [7]—cf. Chap. 13, p. 219.

The Lassy project has extended the available syntactically annotated corpora for Dutch both in size as well as with respect to the various text genres and topical domains. In order to judge the quality of the resources, the annotated corpora have

G. van Noord (✉) · G. Bouma · D. de Kok · J. van der Linde · E. Tjong Kim Sang
University of Groningen, Groningen, The Netherlands
e-mail: g.j.m.van.noord@rug.nl; g.bouma@rug.nl; d.j.a.de.kok@rug.nl; jelmer@ikhoefgeen.nl; e.f.tjong.kim.sang@rug.nl

F. Van Eynde · I. Schuurman · V. Vandeghinste
KU Leuven, Leuven, Belgium
e-mail: frank@ccl.kuleuven.be; ineke.schuurman@ccl.kuleuven.be; vincent@ccl.kuleuven.be

P. Spyns and J. Odijk (eds.), *Essential Speech and Language Technology for Dutch*, Theory and Applications of Natural Language Processing,
DOI 10.1007/978-3-642-30910-6_9,

been externally validated by the Center for Sprogteknologi of the University of Copenhagen. In Sect. 9.5 we present the results of this external validation.

In addition, various browse and search tools for syntactically annotated corpora have been developed and made freely available. Their potential for applications in corpus linguistics and information extraction has been illustrated and evaluated through a series of three case studies.

In this article, we illustrate the potential of the Lassy treebanks by providing a short introduction to the annotations and the available tools, and by describing a number of typical research cases which employ the Lassy treebanks in various ways.

9.2 Annotation and Representation

In this section we describe the annotations in terms of part-of-speech, lemma and syntactic dependency. Furthermore, we illustrate how the annotations are stored in a straightforward XML file format.

Annotations for Lassy Small have been assigned in a semi-automatic manner. Annotations were first assigned automatically, and then our annotators manually inspected these annotations and corrected the mistakes. For part-of-speech and lemma annotation, Tadpole [12] has been used. For the syntactic dependency annotation, we used the Alpino parser [15]. For the correction of syntactic dependency annotations, we employed the TrEd tree editor [8]. In addition, a large number of heuristics have been applied to find annotation mistakes semi-automatically.

The annotations for Lassy Large have been assigned in the same way, except that there has been no manual inspection and correction phase.

9.2.1 Part-of-Speech and Lemma Annotation

The annotations include part-of-speech annotation and lemma annotation for words, and syntactic dependency annotation between words and word groups. Part-of-speech and lemma annotation closely follow the guide-lines developed in D-COI [14]. These guide-lines extend the guide-lines developed in CGN, in order to take into account typical constructs for written language. As an example of the annotations, consider the sentence

(1) In 2005 moest hij stoppen na een meningsverschil met de studio .
In 2005 must he stop after a dispute with the studio .
In 2005, he had to stop after a dispute with the studio

The annotation of this sentence is given here as follows where each line contains the word, followed by the lemma, followed by the part-of-speech tag.

```
In              in              VZ(init)
2005            2005            TW(hoofd,vrij)
```

```
moest            moeten           WW(pv,verl,ev)
hij              hij              VNW(pers,pron,nomin,vol,3,ev,masc)
stoppen          stoppen          WW(inf,vrij,zonder)
na               na               VZ(init)
een              een              LID(onbep,stan,agr)
meningsverschil  meningsverschil  N(soort,ev,basis,onz,stan)
met              met              VZ(init)
de               de               LID(bep,stan,rest)
studio           studio           N(soort,ev,basis,zijd,stan)
.                .                LET()
```

As the example indicates, the annotation not only provides the main part-of-speeches such as VZ (preposition), WW (verb), VNW (pronoun), but also various features to indicate tense, aspect, person, number, gender etc. Below, we describe how the part-of-speech and lemma annotations are included in the XML representation, together with the syntactic dependency annotations.

9.2.2 Syntactic Dependency Annotation

The guide-lines for the syntactic dependency annotation are given in detail in [18]. This manual is a descendent of the CGN and D-Coi syntactic annotation manuals. The CGN syntactic annotation manual [5] has been extended with many more examples and adapted by reducing the amount of linguistic discussions. Some further changes were applied based on the feedback in the validation report of the D-COI project. In a number of documented cases, the annotation guidelines themselves have changed in order to ensure consistency of the annotations, and to facilitate semi-automatic annotation.

Syntactic dependency annotations express three types of syntactic information:

- hierarchical information: which words belong together
- relational information: what is the grammatical function of the various words and word groups (such functions include head, subject, direct object, modifier etc.)
- categorial information: what is the categorial status of the various word groups (categories include NP, PP, SMAIN, SSUB, etc.)

As an example of a syntactic dependency annotation, consider the graph in Fig. 9.1 for the example sentence given above. This example illustrates a number of properties of the representation. Firstly, the left-to-right surface order of the word is not always directly represented in the tree structure, but rather each of the leaf nodes is associated with the position in the string that it occurred at (the subscript). The tree structure does represent hierarchical information, in that words that belong together are represented under one node. Secondly, some word groups are analysed to belong to more than a single word group. We use a co-indexing mechanism to represent such *secondary edges*. In this example, the word `hij` functions both as the subject of the modal verb `moeten` and the main verb `stoppen`—this is indicated by the index 1. Thirdly, we inherit from CGN the practice that punctuation (including sentence internal punctuation) is not analysed syntactically, but simply attached

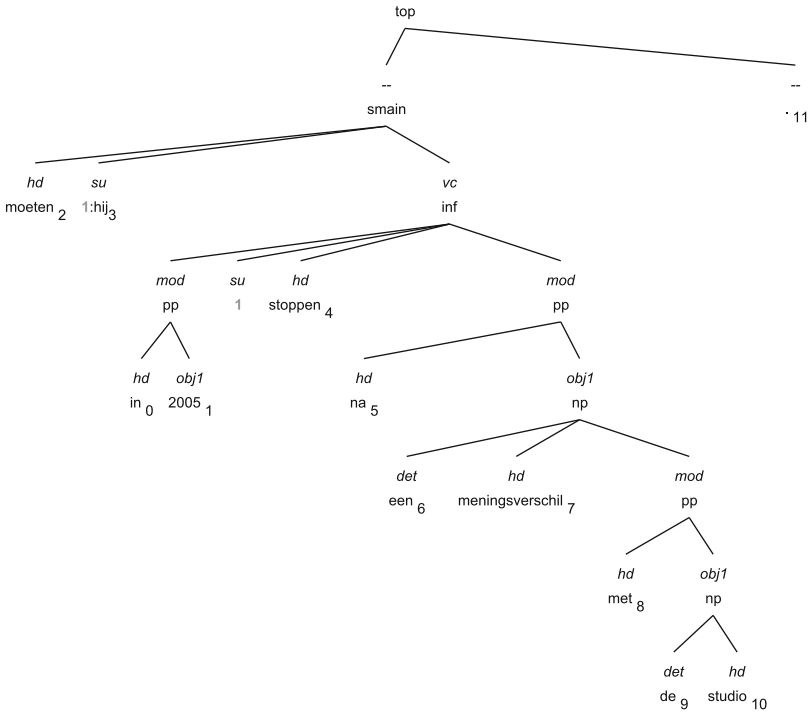

Fig. 9.1 Syntactic dependency graph for sentence (1)

to the top node, to ensure that all tokens of the input are part of the dependency structure. The precise location of punctuation tokens is represented because all tokens are associated with an integer indicating the position in the sentence.

9.2.3 Representation in XML

Both the dependency structures and the part-of-speech and lemma annotations are stored in a single XML format. Advantages of the use of XML include the availability of general purpose search and visualisation software. For instance, we exploit XPath[1] (standard XML query language) to search in large sets of dependency structures, and XQuery to extract information from such large sets of dependency structures.

In the XML-structure, every node is represented by a `node` entity. The other information is presented as values of various XML-attributes of those nodes.

[1] http://www.w3.org/TR/xpath/ and http://www.w3.org/TR/xpath20/

The important attributes are `cat` (syntactic category), `rel` (grammatical function), `postag` (part-of-speech tag), `word`, `lemma`, `index`, `begin` (starting position in the surface string) and `end` (end position in the surface string).

Ignoring some attributes for expository purposes, part of the annotation of our running example is given in XML as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<alpino_ds version="1.3">
  <node begin="0" cat="top" end="12" rel="top">
    <node begin="0" cat="smain" end="11" rel="--">
      <node begin="2" end="3" lemma="moeten" word="moest"/>
      <node begin="3" end="4" index="1" lemma="hij"
            rel="su" word="hij"/>
      <node begin="0" cat="inf" end="11" rel="vc">
        <node begin="0" cat="pp" end="2" rel="mod">
          <node begin="0" end="1" lemma="in"
                rel="hd" word="In"/>
          <node begin="1" end="2" lemma="2005"
                rel="obj1" word="2005"/>
        </node>
        <node begin="3" end="4" index="1" rel="su"/>
        ....
      </node>
    </node>
    <node begin="11" end="12" lemma="." rel="--" word="."/>
  </node>
  <sentence>In 2005 moest hij stoppen na een meningsverschil
             met de studio .</sentence>
</alpino_ds>
```

Leaf nodes have further attributes to represent the part-of-speech tag. The attribute `postag` will be the part-of-speech tag including the various sub-features. The abbreviated part-of-speech tag—the part without the attributes—is available as the value of the attribute `pt`. In addition, each of the sub-features itself is available as the value of further XML attributes. The precise mapping of part-of-speech tags and attributes with values is given in [18]. The actual node for the finite verb `moest` in the example including the attributes to represent the part-of-speech is:

```
<node begin="2" end="3" lemma="moeten" postag="WW(pv,verl,ev)"
      pt="ww" pvagr="ev" pvtijd="verl" rel="hd" word="moest"
      wvorm="pv"/>
```

This somewhat redundant specification of the information encoded in the part-of-speech labels facilitates the construction of queries, since it is possible to refer directly to particular sub-features, and therefore to generalise more easily over part-of-speech labels.

9.3 Querying the Treebanks

As the annotations are represented in XML, there is a variety of tools available to work with the annotations. Such tools include XSLT, XPath and XQuery, as well as a number of special purpose tools—some of which were developed in the course of the Lassy project. Most of these tools have in common that particular parts of the

tree can be identified using the XPath query language. XPath (XML Path Language) is an official W3C standard which provides a language for addressing parts of an XML document. In this section we provide a number of simple examples of the use of XPath to search in the Lassy corpora. We then continue to argue against some perceived limitations of XPath.

9.3.1 Search with XPath

We start by providing a number of simple XPath queries that can be used to search in the Lassy treebanks. We do not give a full introduction to the XPath language—for this purpose there are various resources available on the web.

9.3.1.1 Some Examples

With XPath, we can refer to hierarchical information (encoded by the hierarchical embedding of node elements), grammatical categories and functions (encoded by the `cat` and `rel` attributes), and surface order (encoded by the attributes `begin` and `end`.

As a simple introductory example, the following query:

```
//node[@cat="pp"]
```

identifies all nodes anywhere in a given document, for which the value of the `cat` attribute equals `pp`. In practice, if we use such a query against our Lassy Small corpus using the Dact tool (introduced below), we will get all sentences which contain a prepositional phrase. In addition, these prepositional phrases will be highlighted. In the query we use the double slash notation to indicate that this `node` can appear anywhere in the dependency structure. Conditions about this node can be given between square brackets. Such conditions often refer to particular values of particular attributes. Conditions can be combined using the boolean operators `and`, `or` and `not`. For instance, we can extend the previous query by requiring that the `PP` node should start at the beginning of the sentence:

```
//node[@cat="pp" and @begin="0"]
```

Brackets can be used to indicate the intended structure of the conditions, as in:

```
//node[(@cat="pp" or @cat="advp") and not(@begin="0")]
```

Conditions can also refer to the context of the node. In the following query, we pose further restrictions on a daughter node of the PP category.

```
//node[@cat="pp" and node[@rel="hd" and not(@pt="vz")]]
```

This query will find all sentences in which a PP occurs with a head node for which it is the case that its part-of-speech label is not of the form `VZ(..)`. Such a query will

return quite a few hits—in most cases for prepositional phrases which are headed by multi-word-units such as `in tegenstelling tot` (in contrast with), `met betrekking tot` (with respect to), ... If we want to exclude such multi-word-units, the query could be extended as follows, where we require that there is a `word` attribute, irrespective of its value.

```
//node[@cat="pp" and
       node[@rel="hd" and @word and not(@pt="vz")]]
```

We can look further down inside a node using the single slash notation. For instance, the expression `node[@rel="obj1"]/node[@rel="hd"]` will refer to the head of the direct object. We can also access the value of an attribute of a sub-node as in `node[@rel="hd"]/@postag`.

It is also possible to refer to the mother node of a given node, using the double dot notation. The following query identifies prepositional phrases which are a dependent in a main sentence:

```
//node[@cat="pp" and ../@cat="smain"]
```

Combining the two possibilities we can also refer to sister nodes. In this query, we find prepositional phrases as long as there is a sister which functions as a secondary object:

```
//node[@cat="pp" and ../node[@rel="obj2"]]
```

Finally, the special notation `.//` identifies any node which is embedded anywhere in the current node. The next query finds embedded sentences which include the word `van` anywhere.

```
//node[@cat="ssub" and .//node[@word="van"]]
```

9.3.1.2 Left to Right Ordering

Consider the following example, in which we identify prepositional phrases in which the preposition (the head) is preceded by the NP (which is assigned the `obj1` function). Here we use the operator < to implement *precedence*.

```
//node[@cat="pp" and
       node[@rel="obj1"]/number(@begin)
     < node[@rel="hd"]/number(@begin) ]
```

Note that we use in these examples the `number()` function to map the string value explicitly to a number. This is required in some implementations of XPath.

The operator = can be used to implement *direct precedence*. As another example, consider the problem of finding a prepositional phrase which follows a finite verb directly in a subordinate finite sentence. Initially, we arrive at the following query:

```
//node[@cat="ssub" and
       node[@rel="hd"]/number(@end)
     = node[@cat="pp"]/number(@begin)]
```

This does identify subordinate finite sentences in which the finite verb is directly followed by a PP. But note that the query also requires that this PP is a dependent of the same node. If we want to find a PP anywhere, then the query becomes:

```
//node[@cat="ssub" and
       node[@rel="hd"]/number(@end)
     = //node[@cat="pp"]/number(@begin)]
```

9.3.1.3 Pitfalls

The content and sub-structure of coindexed nodes (to represent secondary edges) is present in the XML structure only once. The index attribute is used to indicate equivalence of the nodes. This may have some unexpected effects. For instance, the following query will *not* match with the dependency structure given in Fig. 9.1.

```
//node[node[@rel="hd" and @lemma="stoppen"] and
       node[@rel="su" and @lemma="hij"]]
```

The reason is, that the subject of `stoppen` itself does not have a subject with `lemma=hij`. Instead, it does have a subject which is co-indexed with a node for which this requirement is true. In order to match this case also, the query should be complicated, for instance as follows:

```
//node[node[@rel="hd" and @lemma="stoppen"] and
       node[@rel="su" and
            ( @lemma="hij" or
              @index=//node[@lemma="hij"]/@index
            )]
```

The example illustrates that the use of co-indexing is not problematic for XPath, but it does complicate the queries in some cases. Some tools (for instance the Dact tool described in Sect. 9.3.3) provide the capacity to define macro substitutions in queries, which simplifies matters considerably.

9.3.2 Comparison with Lai and Bird 2004

In [6] a comparison of a number of existing query languages is presented, by focussing on seven example queries. Here we show that each of the seven queries can be formulated in XPath for the Lassy treebank. In order to do this, we first adapted the queries in a non-essential way. For one thing, some queries refer to English words which we mapped to Dutch words. Some other differences are that there is no (finite) VP in the Lassy treebank. The adapted queries with the implementation in XPath is now given as follows:

1. Find sentences that include the word zag.

   ```
   //node[@word="zag"]
   ```

2. Find sentences that do not include the word zag.

   ```
   not(//node[@word="zag"])
   ```

3. Find noun phrases whose rightmost child is a noun.

```
//node[@cat="np" and node[@pt="n"]/number(@end)=number(@end)]
```

4. Find root sentences that contain a verb immediately followed by a noun phrase that is immediately followed by a prepositional phrase.

```
//node[@cat="smain" and node[@pt="ww"]/number(@end)
    = node[@cat="np"
           and number(@end)
             = //node[@cat="pp"]/number(@begin)
          ]/number(@begin)]
```

5. Find the first common ancestor of sequences of a noun phrase followed by a prepositional phrase.

```
//node[.//node[@cat="np"]/number(@end) =
       .//node[@cat="pp"]/number(@begin) and
   not(node[.//node[@cat="np"]/number(@end) =
            .//node[@cat="pp"]/number(@begin)]) ]
```

6. Find a noun phrase which dominates a word `donker` (dark) that is dominated by an intermediate phrase that is a prepositional phrase.

```
//node[@cat="np" and
      .//node[@cat="pp" and .//node[@word="donker"]]]
```

7. Find a noun phrase dominated by a root sentence. Return the subtree dominated by that noun phrase only.

```
//node[@cat="smain"]/node[@cat="np"]
```

The ease with which the queries can be defined may be surprising to readers familiar with Lai and Bird [6]. In that paper, the authors conclude that XPath is not expressive enough for some queries. As an alternative, the special query language LPATH is introduced, which extends XPath in three ways:

- the additional axis *immediately following*
- the scope operator { . . . }
- the node alignment operators ^ and $

However, we note here that these extensions are unnecessary. As long as the surface order of nodes is explicitly encoded by XML attributes `begin` and `end`, as in the Lassy treebank, then the additional power is redundant. An LPATH query which requires that a node x immediately follows a node y can be encoded in XPath by requiring that the begin-attribute of x equals the end-attribute of y. The examples which motivate the introduction of the other two extensions likewise can be encoded in XPath by means of the begin- and end-attributes. For instance, the LPATH query

```
//node[@cat="smain"]{//node[@cat="np"]$}
```

where an SMAIN node is selected which contains a right-aligned NP can be defined in XPath as:

```
//node[@cat="smain" and number(@end) =
     .//node[@cat="np"]/number(@end)]
```

Based on these examples we conclude that there is no motivation for an ad-hoc special purpose extension of XPath, but that instead we can safely continue to use the XPath standard.

9.3.3 A Graphical User Interface for Lassy

Dact is a recent easy-to-use open-source tool, available for multiple platforms, to browse and search through Lassy treebanks. It provides graphical tree visualizations of the dependency structures of the treebank, full XPath search to select relevant dependency structures in a given corpus and to highlight the selected nodes of dependency structures, simple statistical operations to generate frequency lists for any attributes of selected nodes, and sentence-based outputs in several formats to display selected nodes e.g. by bracketing the selected nodes, or by a keyword-in-context presentation. Dact can be downloaded from http://rug-compling.github.com/dact/.

For the XML processing, Dact supports both the libxml2 (http://xmlsoft.org) and the Oracle Berkeley DB XML (http://www.oracle.com) libraries. In the latter case, database technology is used to preprocess the corpus for faster query evaluation. In addition, the use of XPath 2.0 is supported. Furthermore, Dact provides macro expansion in XPath queries.

The availability of XPath 2.0 is useful in order to specify quantified queries (argued for in the context of the Lassy treebanks in [1]). As an example, consider the query in which we want to identify a NP which contains a VC complement (infinite VP complement), in such a way that there is a noun which is preceded by the head of that NP, and which precedes the VC complement. In other words, in such a case there is an (extraposed) VC complement of a noun for which there is another noun which appears in between the noun and the VC complement. The query can be formulated as:

```
//node[@cat="np" and
      ( some $interm in //node[@pos="noun"]
   satisfies (   $interm/number(@begin)
               < node[@rel="vc"]/number(@begin) and
                 $interm/number(@end)
               > node[@rel="hd"]/number(@end)
             ))]
```

The availability of a macro facility is useful to build up more complicated queries in a transparent way. The following example illustrates this point. Macro's are defined using the format `name = string`. A macro is used by putting the name between `% %`. The following set of macro's defines the solution to the fifth problem posed in [6] in a more transparent manner. In order to define the minimal node which dominates a NP PP sequence, we first define the notion *dominates a NP PP sequence*, and then use it to state that the first common ancestor of a sequence of NP

PP is a node which is an ancestor of a NP PP sequence, but which does not contain a node which is an ancestor of a NP PP sequence.

```
b = number(@begin)
e = number(@end)
dominates_np_pp_seq =
  .//node[@cat="np"]/%e% = .//node[@cat="pp"]/%b%
q5 = //node[%dominates_np_pp_seq% and
            not(node[%dominates_np_pp_seq%])]
```

9.4 Using the Lassy Treebanks

9.4.1 Introduction

The availability of manually constructed treebanks for research and development in natural language processing is crucial, in particular for training statistical syntactic analysers or statistical components of syntactic analysers of a more hybrid nature. In addition such high quality treebanks are important for evaluation purposes for any kind of automatic syntactic analysers.

Syntactically annotated corpora of the size of Lassy Small are also very useful resources for corpus linguists. Note that the size of Lassy Small (one million words) is the same as the subset of the Corpus of Spoken Dutch (CGN) which has been syntactically annotated. Furthermore, the syntactic annotations of the CGN are also available in a format which is understood by Dact. This implies that it is now straightforward to perform corpus linguistic research both on spoken and written Dutch. Below, we provide a simple case study where we compare the frequency of WH-questions formed with `wie` (who) as opposed to `welk(e)` (which).

It is less obvious whether large quantity, lower quality treebanks are a useful resource. As one case in point, we note that a preliminary version of the Lassy Large treebank was used as *gold standard* training data to train a memory-based parser for Dutch [12]. In this article, we illustrate the expected quality of the automatic annotations, and we discuss an example study which illustrates the promise of large quantity, lower quality treebanks. In this section, we therefore focus on the use of the Lassy Large treebank.

9.4.2 Estimating the Frequency of Question Types

As an example of the use of Lassy Small, we report on a question of a researcher in psycholinguistics who focuses on the linguistic processing of WH-questions from a behavioral (e.g. self-paced reading studies) and neurological (event-related potentials) viewpoint. She studies the effect of information load: the difference between `wie` and `welk(e)` in for example:

Table 9.1 number of hits per query

	CGN		Lassy	
	#	%	#	%
wie	201	57.9	61	64.2
welk(e)	146	42.1	34	35.8

(2) Wie bakt het lekkerste brood?
Who bakes the nicest bread?

(3) Welke bakker bakt het lekkerste brood?
Which baker bakes the nicest bread?

To be sure that the results she finds are psycholinguistic or neurolinguistic in nature, she wants to be able to compare them to a frequency count in corpora.

Such questions can now be answered using the Lassy Small treebank or the CGN treebank by posing two simple queries. The following query finds WH-questions formed with `wie`:

```
//node[@cat="whq" and node[@rel="whd" and
      ( @lemma="wie" or .//node[@lemma="wie"] )]]
```

The number of hits of the queries are given in Table 9.1:

9.4.3 Estimation of the Quality of Lassy Large

In order to judge the quality of the Lassy Large corpus, we evaluate the automatic parser that was used to construct Lassy Large on the manually verified annotations of Lassy Small. The Lassy Small corpus is composed of a number of sub-corpora. Each sub-corpus is composed of a number of documents. In the experiment, Alpino (version of October 1, 2010) was applied to a single document, using the same options which have been used for the construction of the Lassy Large corpus. With these options, the parser delivers a single parse, which it believes is the best parse according to a variety of heuristics. These include the disambiguation model and various optimizations of the parser presented in [9, 16, 17]. Furthermore, a time-out is enforced in order that the parser cannot spend more than 190 s on a single sentence. If no result is obtained within this time, the parser is assumed to have returned an empty set of dependencies, and hence such cases have a very bad impact on accuracy.

In the presentation of the results, we aggregate over sub-corpora. The various `dpc-` sub-corpora are taken from the Dutch Parallel Corpus, and meta-information should be obtained from that corpus. The various `WR-` and `WS` corpora are inherited from D-COI. The `wiki-` subcorpus contains wikipedia articles, in many cases about topics related to Flanders.

Parsing results are listed in Table 9.2. Mean accuracy is given in terms of the f-score of named dependency relations. As can be observed from this table, parsing accuracies are fairly stable across the various sub-corpora. An outlier is the result of the parser on the WR-P-P-G sub-corpus (legal texts), both in terms of

Table 9.2 Parsing results (f-score of named dependencies) on the Lassy Small sub-corpora

Name	f-score	ms	#sent	Length	Name	f-score	ms	#sent	Length
dpc-bal-	92.77	1,668	620	14.2	dpc-bmm-	87.81	4,096	794	19.6
dpc-cam-	91.78	2,913	508	19.6	dpc-dns-	90.48	1,123	264	14.5
dpc-eli-	89.81	4,453	603	18.8	dpc-eup-	89.88	8,642	233	26.1
dpc-fsz-	85.74	4,492	574	19.1	dpc-gaz-	88.51	3,410	210	18.1
dpc-ibm-	90.13	4,753	419	20.2	dpc-ind-	91.14	4,010	1,650	20.6
dpc-kam-	89.82	4,671	52	25.6	dpc-kok-	88.00	2,546	101	18.3
dpc-med-	90.28	3,906	650	20.9	dpc-qty-	89.86	7,044	618	22.2
dpc-riz-	86.61	4,926	210	20.1	dpc-rou-	91.50	2,218	1,356	16.7
dpc-svb-	89.69	1,939	478	15.8	dpc-vhs-	90.83	1,819	461	14.4
dpc-vla-	90.57	2,545	1,915	16.8	wiki	88.85	1,940	7,341	13.4
WR-P-E-C	84.77	1,827	1,014	12.1	WR-P-E-E	82.61	3,599	90	20.1
WR-P-E-H	88.10	2,110	2,832	11.4	WR-P-E-I	87.78	4,051	9,785	20.4
WR-P-E-J	87.69	5,276	699	21.5	WR-P-P-B	92.07	318	275	7.3
WR-P-P-C	88.08	2,089	5,648	14.8	WR-P-P-E	89.14	3,759	306	19.0
WR-P-P-F	83.11	4,362	397	16.4	WR-P-P-G	80.32	10,410	279	23.2
WR-P-P-H	91.42	2,109	2,267	16.4	WR-P-P-I	90.43	3,369	5,789	20.0
WR-P-P-J	86.79	6,278	1,264	23.8	WR-P-P-K	89.37	3,715	351	19.9
WR-P-P-L	88.70	3,406	1,115	18.5	WS	90.40	1,596	14,032	14.7
Total	89.17	2,819	65,200	16.8					

accuracy and in terms of parsing times. We note that the parser performs best on the dpc-bal- subcorpus, a series of speeches by former prime-minister Balkenende.

9.4.4 The Distribution of zelf and zichzelf

As a further example of the use of parsed corpora to further linguistic insights, we consider a recent study [2] of the distribution of weak and strong reflexive objects in Dutch.

If a verb is used reflexively in Dutch, two forms of the reflexive pronoun are available. This is illustrated for the third person form in the examples below.

(4) Brouwers schaamt **zich/*zichzelf** voor zijn schrijverschap.
Brouwers shames *self1/self2* for his writing
Brouwers is ashamed of his writing

(5) Duitsland volgt ***zich/zichzelf** niet op als Europees kampioen.
Germany follows *self1/self2* not PART as European Champion
Germany does not succeed itself as European champion

(6) Wie **zich/zichzelf** niet juist introduceert, valt af.
Who self1/self2 not properly introduces, is out
Everyone who does not introduce himself properly, is out.

The choice between *zich* and *zichzelf* depends on the verb. Generally three groups of verbs are distinguished. Inherent reflexives never occur with a non-reflexive argument and occur only with *zich*(4). Non-reflexive verbs seldom, if ever occur with a reflexive argument. If they do, however, they can only take *zichzelf* as a reflexive argument (5). Accidental reflexives can be used with both *zich* and *zichzelf*, (6). Accidental reflexive verbs vary widely as to the frequency with which they occur with both arguments. [2] set out to explain this distribution.

The influential theory of [10] explains the distribution as the surface realization of two different ways of reflexive coding. An accidental reflexive that can be realized with both *zich* and *zichzelf* is actually ambiguous between an inherent reflexive and an accidental reflexive (which always is realized with *zichzelf*). An alternative approach is that of [3, 4, 11], who have claimed that the distribution of weak vs. strong reflexive object pronouns correlates with the proportion of events described by the verb that are self-directed vs. other-directed.

In the course of this investigation, a first interesting observation is, that many inherently reflexive verbs, which are claimed not to occur with *zichzelf*, actually often do combine with this pronoun. Two typical examples are:

(7) Nederland moet stoppen zichzelf op de borst te slaan
Netherlands must stop self2 on the chest to beat
The Netherlands must stop beating itself on the chest

(8) Hunze wil zichzelf niet al te zeer op de borst kloppen
Hunze want self2 not all too much on the chest knock
Hunze doesn't want to knock itself on the chest too much

With regards to the main hypothesis of their study, Bouma and Spenader [2] use linear regression to determine the correlation between reflexive use of a (non-inherently reflexive) verb and the relative preference for a weak or strong reflexive pronoun. Frequency counts are collected from the parsed TwNC corpus (almost 500 million words). They limit the analysis to verbs that occur at least 10 times with a reflexive meaning and at least 50 times in total, distinguishing uses by subcategorization frames. The statistical analysis shows a significant correlation, which accounts for 30 % of the variance of the ratio of nonreflexive over reflexive uses.

9.5 Validation

The Lassy Small and Lassy Large treebanks have been validated by a project-external body, the Center for Sprogteknologi, University of Copenhagen. The validation report gives a detailed account of the validation of the linguistic annotations of syntax, PoS and lemma in the Lassy treebanks. The validation comprises extraction of validation samples, manual checks of the content, and computation of named dependency accuracy figures of the syntax validation results.

The content validation falls in two parts: validation of the linguistic annotations (PoS-tagging, lemmatization) and the validation of the syntactic annotations. The validation of the syntactic annotation was carried out on 250 sentences from Lassy Large and 500 sentences from Lassy Small, all randomly selected. The validation of the lemma and PoS-tag annotations was carried out on the same sample from Lassy Small as for syntax, i.e. 500 sentences.

Formal validation i.e. the checking of formal information such as file structure, size of files and directories, names of files etc. is not included in this validation task but no problems were encountered in accessing the data and understanding the structure. For the syntax, the validators computed a sentence based accuracy (number of sentences without errors divided by the total number of sentences). For Lassy Large, the validators found that the syntactic analysis was correct for a proportion of 78.4 % of the sentences. For Lassy Small, the proportion of correct syntactic analyses was 97.8 %. Out of the 500 validated sentences with a total of 8,494 words, the validators found 31 words with a wrong lemma (the accuracy of the lemma annotation therefore is 99.6 %. For this same set of sentences, validators found 116 words with wrong part-of-speech tag (accuracy 98.63 %).

In conclusion, the validation states that the Lassy corpora comprise a well elaborated resource of high quality. Lassy Small, the manually verified corpus, has really fine results for both syntax, part-of-speech and lemma, and the overall impression is very good. Lassy Large also has fine results for the syntax. The overall impression of the Lassy Large annotations is that the parser succeeds in building up acceptable trees for most of the sentences. Often the errors are merely a question of the correct labeling of the nodes.

9.6 Conclusion

In this article we have introduced the Lassy treebanks, and we illustrated the lemma, part-of-speech and dependency annotations. The quality of the annotations has been confirmed by an external validation. We provided evidence that the use of the standard XPath language suffices for the identification of relevant nodes in the treebanks, countering some evidence to the contrary by Lai and Bird [6] and Bouma [1]. We illustrated the potential usefulness of Lassy Small by estimating the frequency of question types in the context of a psycho-linguistic study. We furthermore illustrated the use of the Lassy Large treebank in a study of the distribution of the two Dutch reflexive pronouns `zich` and `zichzelf`.

Appendices

A. List of Category Labels

Label	Explanation	Label	Explanation
AP	Adjectival phrase	ADVP	Adverbial phrase
AHI	*aan het*-infinitive VP	CONJ	Conjunction
CP	Subordinate sentence	DETP	Determiner phrase
DU	Discourse unit	INF	Bare infinitive VP
NP	Noun phrase	OTI	*om te*-infinitive VP
PPART	Past participle VP	PP	Prepositional phrase
PPRES	Present participle VP	REL	Relative clause
SMAIN	Root sentence	SSUB	Subordinate finite VP
SVAN	*van* finite VP	SV1	Verb-first phrase (yes/no question, imperative)
TI	*te*-infinitive VP	WHREL	Free relative clause
WHSUB	Embedded question	WHQ	WH-question

B. List of Dependency Labels

Label	Explanation	Label	Explanation
APP	Apposition		
BODY	Body	CMP	Complementizer
CNJ	Conjunct	CRD	Coordinator
DET	Determiner	DLINK	Discourse-link
DP	Discourse-part	HD	Head
HDF	Post-position	LD	Locative/directional complement
ME	Measure complement	MOD	Modifier
MWP	Part of multi-word-unit	NUCL	Nucleus
OBCOMP	Complement of comparison	OBJ1	Direct object
OBJ2	Secondary object	PC	Prepositional complement
POBJ1	Temporary direct object	PREDC	Predicative complement
PREDM	Predicative modifier	RHD	Head of relative clause
SAT	Satellite	SE	Reflexive object
SU	Subject	SUP	Temporary subject
SVP	Particle	TAG	Tag
VC	Verbal complement	WHD	Head of WH-question

References

1. Bouma, G.: Starting a Sentence in Dutch. Ph.D. thesis, University of Groningen (2008)
2. Bouma, G., Spenader, J.: The distribution of weak and strong object reflexives in Dutch. In: van Eynde, F., Frank, A., Smedt, K.D., van Noord, G. (eds.) Proceedings of the Seventh International Workshop on Treebanks and Linguistic Theories (TLT 7), no. 12 in LOT Occasional Series, pp. 103–114. Netherlands Graduate School of Linguistics, Utrecht, The Netherlands (2009)
3. Haspelmath, M.: A frequentist explanation of some universals of reflexive marking (2004). Draft of a paper presented at the Workshop on Reciprocals and Reflexives, Berlin
4. Hendriks, P., Spenader, J., Smits, E.J.: Frequency-based constraints on reflexive forms in Dutch. In: Proceedings of the 5th International Workshop on Constraints and Language Processing, pp. 33–47. Roskilde, Denmark (2008). http://www.ruc.dk/dat_en/research/reports
5. Hoekstra, H., Moortgat, M., Schouppe, M., Schuurman, I., van der Wouden, T.: CGN Syntactische Annotatie (2004). http://www.tst-centrale.org/images/stories/producten/documentatie/cgn_website/doc_Dutch/topics/annot/syntax/syn_prot.pdf
6. Lai, C., Bird, S.: Querying and updating treebanks: a critical survey and requirements analysis. In: In Proceedings of the Australasian Language Technology Workshop, pp. 139–146. Sydney, Australia (2004)
7. Oostdijk, N., Reynaert, M., Monachesi, P., van Noord, G., Ordelman, R., Schuurman, I., Vandeghinste, V.: From D-Coi to SoNaR. In: Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC 2008), Marrakech, Morocco (2008)
8. Pajas, P., Štěpánek, J.: Recent advances in a feature-rich framework for treebank annotation. In: Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008), pp. 673–680. Coling 2008 Organizing Committee, Manchester, UK (2008). http://www.aclweb.org/anthology/C08-1085
9. Prins, R., van Noord, G.: Reinforcing parser preferences through tagging. Traitement Automatique des Langues **44**(3), 121–139 (2003)
10. Reinhart, T., Reuland, E.: Reflexivity. Linguist. Inq. **24**, 656–720 (1993)
11. Smits, E.J., Hendriks, P., Spenader, J.: Using very large parsed corpora and judgement data to classify verb reflexivity. In: Branco, A. (ed.) Anaphora: Analysis, Algorithms and Applications, pp. 77–93. Springer, Berlin (2007)
12. van den Bosch, A., Busser, B., Canisius, S., Daelemans, W.: An efficient memory-based morphosyntactic tagger and parser for Dutch. In: Dirix,P., Schuurman, I., Vandeghinste, V., van Eynde, F. (eds.) Computational Linguistics in the Netherlands 2006. Selected Papers from The Seventeenth CLIN meeting, LOT Occassional Series, pp. 99–114. LOT Netherlands Graduate School of Linguistics, Utrecht, The Netherlands. Leuven, Belgium (2007)
13. van Eerten, L.: Over het Corpus Gesproken Nederlands. Nederlandse Taalkunde **12**(3), 194–215 (1997)
14. Van Eynde, F.: Part Of Speech Tagging En Lemmatisering Van Het D-Coi Corpus (2005). http://www.let.rug.nl/~vannoord/Lassy/POS_manual.pdf
15. van Noord, G.: **A**t **L**ast **P**arsing **I**s **N**ow **O**perational. In: TALN 2006 Verbum Ex Machina, Actes De La 13e Conference sur Le Traitement Automatique des Langues naturelles, Leuven, pp. 20–42 (2006)
16. van Noord, G.: Learning efficient parsing. In: EACL 2009, The 12th Conference of the European Chapter of the Association for Computational Linguistics, Athens, Greece, pp. 817–825 (2009)

17. van Noord, G., Malouf, R.: Wide coverage parsing with stochastic attribute value grammars (2005). Draft available from the authors. A preliminary version of this paper was published in the Proceedings of the IJCNLP workshop Beyond Shallow Analyses, Hainan, China (2004)
18. van Noord, G., Schuurman, I., Bouma, G.: Lassy syntactische annotatie, revision 19455 (2011). http://www.let.rug.nl/vannoord/Lassy/sa-man_lassy.pdf