Using a Treebank to study Weak and Strong Object Reflexives in Dutch
LASSY case study WP 6.2

Gosse Bouma
Information Science
University of Groningen

September, 2009

Abstract

In this case study, we explain how to use the LASSY treebank for studying the distribution of weak and strong object reflexives in Dutch. The results of this work have been published elsewhere as Bouma and Spenader (2009). Here we concentrate on the technical aspects of gathering the data, and some of the statistical manipulation we did on the basis of this.

The linguistic motivation for carrying out this research is a conjecture by Haspelmath, which says that the probability of using a strong reflexive increases if the governing verb is a verb that in general has a strong tendency to be used non-reflexively. To study this hypothesis, we need counts for the number of times a verb is used reflexively and non-reflexively, and, for the reflexive uses, counts for the number of times a weak and a strong reflexive pronoun is used. As will become clear below, various subtleties play a role in deciding what counts as a relevant case of non-reflexive and reflexive use, and in deciding how to distinguish verbs (i.e. on the basis of form or meaning).

We start with an overview of the research question in Bouma and Spenader (2009). Next, we explain how the relevant data collection was carried out, on the basis the LASSY Large Corpus, a large newspaper corpus that was automatically annotated (by Alpino) with syntactic dependency trees. The data was analyzed using the statistics package R. We also document this part of the research. Finally, we give the results of Bouma and Spenader (2009).

1 Explaining the distribution of weak and strong reflexives

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.

(1) a. Brouwers schaamt **zich**/*__zichzelf__ voor zijn schrijverschap.
Brouwers is ashamed of his writing
b. Duitsland volgt **zichzelf** niet op als Europees kampioen.
Germany does not succeed itself as European champion
c. Wie **zich**/__zichzelf__ niet juist introduceert, valt af.
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 are claimed to never occur with a non-reflexive argument, and as a reflexive argument are claimed to use *zich* exclusively, (1a). Non-reflexive verbs seldom, if ever occur with a reflexive argument. If they do however, they can only take *zichzelf* as a reflexive argument (1b). Accidental reflexives can be used with both *zich* and *zichzelf*, (1c). Accidental reflexive verbs vary widely as to the frequency with which they occur with both arguments and it is this distribution that we would like to explain.

What exactly governs the choice between the weak and strong forms of a reflexive in the case of accidental reflexive verbs is largely unclear. Haspelmath (2004), Smits, Hendriks, and Spenader (2007), and Hendriks, Spenader, and Smits (2008) have claimed that the distribution of weak vs. strong reflexive object pronouns (i.e. reflexives that are the object of a verb) correlates with the proportion of events described by the verb that are self-directed vs. other-directed. The claim is that if a verb is rarely used to express self-directed events, there will be a tendency to use the strong reflexive form when it is used reflexively to signal this marked use of the verb. The assumption behind the claim is that when the expectation that a given action will be self-directed is weak, emphasis on the reflexive argument is preferred, so the strong reflexive is used. Such emphasis is less likely if the verb is used with a self-directed meaning relatively often, and therefore the weak reflexive, which is shorter and should otherwise always be preferable, will be sufficient. This is in line with the claim that inherent reflexives only occur with weak reflexives, since they only occur with reflexive meaning.[1]

Our research builds upon the work in Smits, Hendriks, and Spenader (2007) and Hendriks, Spenader, and Smits (2008), who studied the distribution of reflexive vs. nonreflexive use and the choice for a weak or strong form for 45 Dutch transitive verbs. Smits, Hendriks, and Spenader (2007) found a linear correlation between reflexive and non-reflexive usage (counting all third person NPs) for 21 % of the data in an 80 M word corpus (parsed using Alpino) for the verbs sufficiently frequent in the corpus. By combining this with judgement data, they were able to obtain 83% correlation. Hendriks, Spenader, and Smits (2008), using a 300 M word corpus and 32 verbs obtained a correlation of 28% and a correlation of 30% when first and second person reflexives were included. Haspelmath (2004) suggests that only the ratio of pronominal objects to reflexive objects is relevant for determining the degree to which a verb is introverted (tends to describe self-directed events) or extroverted (tends to describe other-directed events). Hendriks, Spenader, and Smits (2008) found that the model proposed by Haspelmath yielded a correlation of 45%.

The research reported in Bouma and Spenader (2009) differs from the approach of Hendriks, Spenader, and Smits (2008) in that it takes into account all transitive verbs in the corpus, and then uses this very large set to test the different models of reflexive choice. The larger set of verbs gives a more complete picture, but also requires a fully automatic method for data collection. The techniques for doing this are described below.

Below, we first introduce the LASSY XML format for dependency trees and some of the general tools for working with this data. Next, we move to the use of XQuery, the XML query language that was used to do the data collection. We conclude with a section on the use of R for statistical analysis of the data.

[1] Note however that many inherent reflexives, like *zich herinneren*, (to remember) or *zich verspreiden*, (to spread out), can't really be characterized as being self-directed actions because the reflexive object does not seem to have a thematic role.

2 LASSY Dependency Trees in XML

Sentences in the LASSY corpus are parsed using the Alpino parser, and then stored as XML. Only a small part of the corpus (LASSY small) has been manually corrected. Figure 1 shows the dependency graph for the example in (2) as it is stored in XML. The dependency relations can also be visually displayed as shown in figure 2.[2] The tool **dtview** can be used to display dependency trees on the screen, as shown in figure 3.

(2) Dat komt doordat Nederlandse reders Belgische schepen kopen
That is because Dutch ship-owners Belgian ships buy

Nodes can be selected and highlighted by means of the highlight menu option in dtview. As dtview is visualizing an underlying XML document, the language XPath[3] (for locating elements in XML documents) is used for making selections. To select verbs, for instance, one may use the XPath expression in (3a) below. To select verbs which have the subcategorization-property of being transitive, one may use (3b). Alternatively, to select all verbs occurring with a direct object (this class is actually quite a bit larger than the verb with a `sc=transitive` attribute, as verbs may select additional complements as well), one may use (3c). The example in figure 4 illustrates the effect of highlighting.

(3) a. `//node[@pos="verb"]`
b. `//node[@pos="verb" and @sc="transitive"]`
c. `//node[@pos="verb" and ../node[@rel="obj1"]]`

The tool `dtsearch` can be used to search LASSY corpora automatically, for nodes matching a given XPath expression. To find all verbs that co-occur with *zich* as direct object, one might use the dtsearch command below. The `-s` flag ensures that matching sentences are displayed, with the matching word or constituent in square brackets.

(4)
```
dtsearch -s -q '//node[@pos="verb" and ../node[@rel="obj1" and @root="zich"]]'
AD1999
```

Sample output is:

```
AD1999/ad19990104/ad19990104-45-8-4.xml
       Hij [dringt] zich immers niet op de voorgrond , maar naar de voorpagina .
AD1999/ad19990104/ad19990104-111-14-1.xml
       De afstand [perst] zich samen .
AD1999/ad19990104/ad19990104-117-4-4.xml
       Daarom [sluiten] andere regio's zich wellicht bij een staking aan .
AD1999/ad19990105/ad19990105-2-3-2.xml
        " Ook hier ging het om aanvallen op mensen , die zich niet konden [verdedigen] .
AD1999/ad19990105/ad19990105-36-4-3.xml
        De Waal [drukte] zich voorzichtiger uit : " Er is reden voor enige argwaan . "
AD1999/ad19990105/ad19990105-78-1-3.xml
        De EO mag zich voor het eerst de grootste van Nederland [noemen].
```

[2] This example was constructed using automatic conversion of the XML into a Latex-file, on the basis of which a picture in pdf was generated.

[3] See `http://www.w3.org/TR/xpath20/`. Note that the tools `dtview` and `dtsearch` use XPath 1.0, in which some of the functionality of XPath 2.0 is missing (esp. support for regular expressions).

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<alpino_ds version="1.2">
  <node begin="0" cat="top" end="9" id="0" rel="top">
    <node begin="0" cat="smain" end="8" id="1" rel="--">
      <node begin="0" end="1" frame="determiner(het,nwh,nmod,pro,nparg)" id="2" infl="het"
        lcat="np" pos="det" rel="su" root="dat" wh="nwh" word="Dat"/>
      <node begin="1" end="2" frame="verb(zijn,sg3,intransitive)" id="3" infl="sg3"
        lcat="smain" pos="verb" rel="hd" root="kom" sc="intransitive" word="komt"/>
      <node begin="2" cat="cp" end="8" id="4" rel="mod">
        <node begin="2" end="3" frame="complementizer" id="5" lcat="cp" pos="comp"
          rel="cmp" root="doordat" word="doordat"/>
        <node begin="3" cat="ssub" end="8" id="6" rel="body">
          <node begin="3" cat="np" end="5" id="7" rel="su">
            <node begin="3" end="4" frame="adjective(e)" id="8" infl="e" lcat="ap" pos="adj"
             rel="mod" root="Nederlands" word="Nederlandse"/>
            <node begin="4" end="5" frame="noun(de,count,pl)" gen="de" id="9" lcat="np"
              num="pl" pos="noun" rel="hd" root="reder" word="reders"/>
          </node>
          <node begin="5" cat="np" end="7" id="10" rel="obj1">
            <node begin="5" end="6" frame="adjective(e)" id="11" infl="e" lcat="ap" pos="adj"
             rel="mod" root="Belgisch" word="Belgische"/>
            <node begin="6" end="7" frame="noun(het,count,pl)" gen="het" id="12" lcat="np"
               num="pl" pos="noun" rel="hd" root="schip" word="schepen"/>
          </node>
          <node begin="7" end="8" frame="verb(hebben,pl,transitive)" id="13" infl="pl"
            lcat="ssub" pos="verb" rel="hd" root="koop" sc="transitive" word="kopen"/>
        </node>
      </node>
    </node>
    <node begin="8" end="9" frame="punct(punt)" id="14" lcat="punct" pos="punct" rel="--"
        root="." special="punt" word="."/>
  </node>
  <sentence>Dat komt doordat Nederlandse reders Belgische schepen kopen .</sentence>
  <comments>
    <comment>Q#ad19990104-1-2-4|Dat komt doordat Nederlandse reders Belgische schepen kopen
.|1|1|-0.10577161400000001</comment>
  </comments>
</alpino_ds>
```

Figure 1: LASSY XML

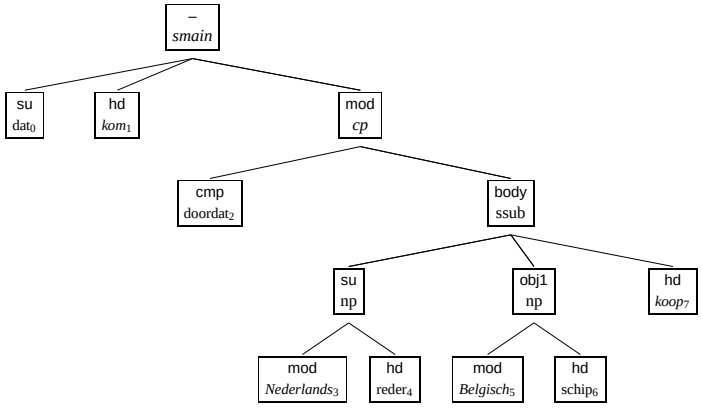

Figure 2: Dependency tree representation of the XML in figure 1

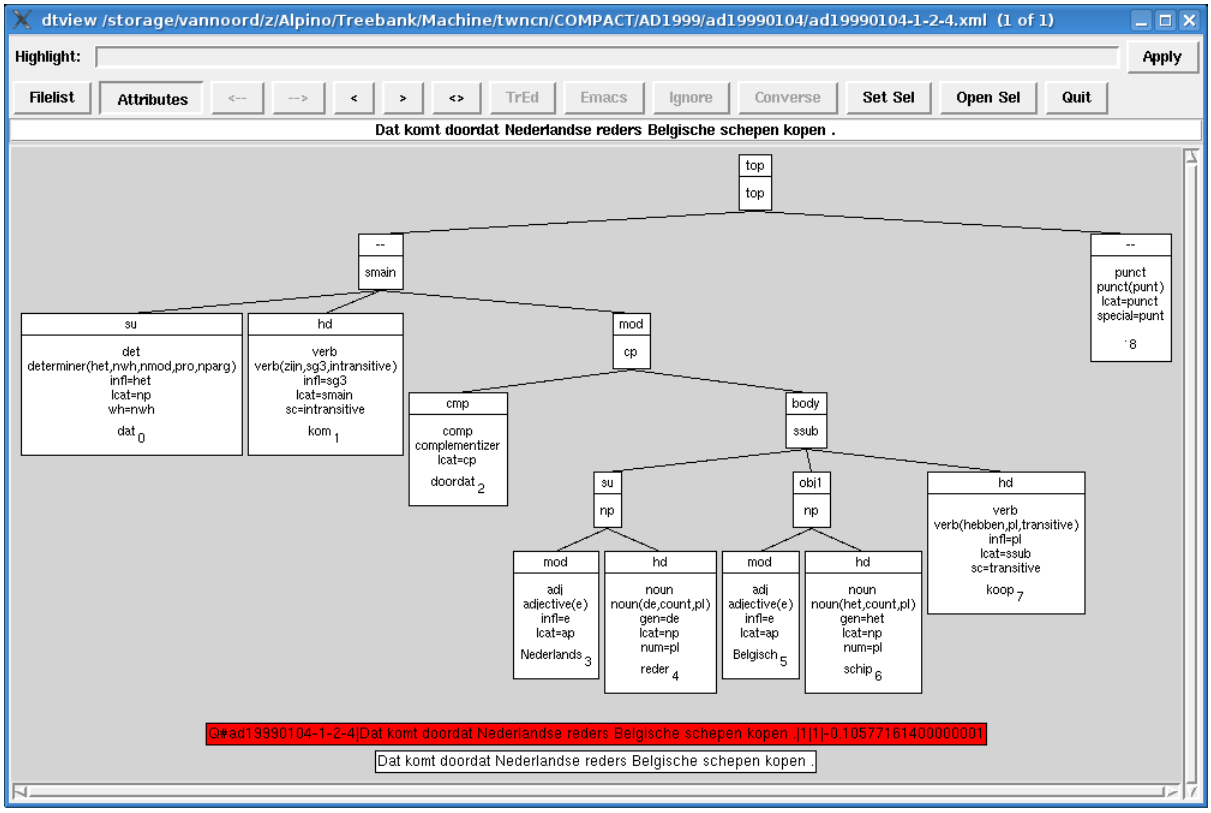

Figure 3: Using dtview to display dependency trees. Selecting the **attributes** button gives additional information on nodes in the tree.

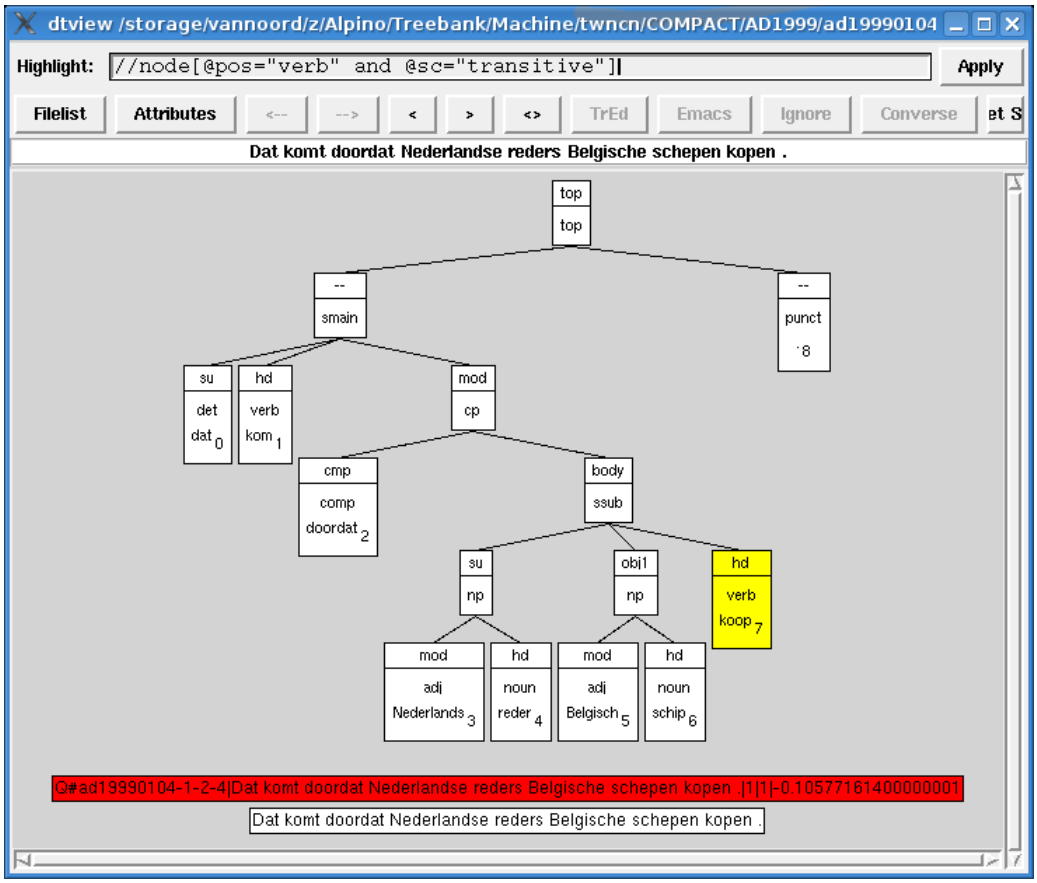

Figure 4: Using the highlight option in dtview to select nodes in a tree.

684	houd	104	verzeker	49	herken
463	laat	100	heb	43	waan
281	noem	90	voeg	43	verplicht
271	bereid_voor	88	dien_aan	42	laat_uit
211	sluit_aan	83	leid	41	sla
177	zie	81	inspireer	40	behandel
158	stel	68	breid_uit	37	voel
120	presenteer	66	versterk	36	meld
112	meld_af	56	verleid	35	maak
111	profileer	54	verdedig	34	breng

Table 1: The number of times a verb occurs with zich as direct object in AD1999. Only the 30 most fequent verbs are shown.

Note that as a means of gathering data for our research question, this is not a perfect output format, as sentences instead of verbs are returned, and also, inflected verb forms. We should count root forms, as distinctions in inflection are most likely not relevant for the linguistic question we want to investigate. Fortunately, `dtsearch` also has an option -r, which returns statistics for root forms that satisfy a given query. A sample of the output of the query in (5) is given in table 1.

(5)
```
dtsearch -r -q '//node[@pos="verb" and ../node[@rel="obj1" and @root="zich"]]'
AD1999
```

In principle, we could use `dtsearch` to gather data for our research question. All we need to do is search the corpus for verbs occurring with an arbitrary direct object, with *zich* as direct object, and with *zichzelf* as direct object. A disadvantage of this method is that it is not very efficient: the whole corpus has to be processed three times, to collect statistics on three different search queries.[4] Another disadvantage is that the granularity of results is limited to what has been built into dtsearch. Below, we will argue, for instance, that it is better to collect counts for combinations of a root form and a subcategorization frame. I.e. we want to distinguish between `node[@root="prijs" and @sc="transitive"]` (*zij prees de Amsterdamse kliniek*) and `node[@root="prijs" and @sc="als_pred_np"]` (*de media prijzen Bush als een begenadigd leerling van Clinton*) This is not possible in dtsearch as it stands. In general, dtsearch is limited to counting the number of sentences, words, or root forms that satisfy a given XPath expression. It does not have functionality to return combinations of words, root forms, or grammatical features in nodes satisfying a given query. Therefore, we now turn to the use of XQuery, an XML query language which is more powerful than dtsearch.

3 Using XQuery

XQuery[5] is a query language for XML. XQuery can be used to locate elements in an XML document, and to return arbitrary parts of that element (i.e. attribute values or embedded

[4]Some optimization can be achieved by combining dtsearch with an IR-index that first selects sentences that could match the query (i.e. only sentences containing *zichzelf* for the third query).

[5]`http://www.w3.org/TR/xquery`

```
<results>
  { for $node in
        collection('ad19990104')//node[@pos="verb"
                                       and ../node[@rel="obj1" and @root="zich"] ]
    return
         <verb>
             {$node/@root}
             {$node/@sc}
         </verb>
   }
</results>
```

Figure 5: XQuery script for extracting verbs selecting *zich* as direct object and returning the value of the `root` and `sc` attributes as part of a `verb` element.

elements) or its context. As it is a programming language, it has much more functionality than XPath (which is part of XQuery). It is ideally suited for information extraction tasks.[6]

The example in figure 5 identifies the same nodes as the dtsearch examples shown in the previous section. The `for` clause selects the relevant nodes. Each match instantiates the variable `$node` as the matching XML element. The `return` statement specifies what to return. In this case, a `verb` element is returned, and the two attributes `root` and `sc` of the current $node are added as attributes to the verb element. The example assumes that there is a directory `ad19990104` which contains XML documents (Alpino/LASSY dependency trees in our case). As we want our output to be valid XML (for later processing, for instance) the output of the `for` loop (a series of `verb` elements) is enclosed in a `results` element.

To execute the script, one can use an XQuery processor such as saxon.[7] The result of running the script is shown in figure 6.

3.1 Why include the `sc` attribute?

We distinguish verbs by their subcategorization frame, as a first step towards distinguishing word senses. A problem for automatic data collection is that most verbs are extremely ambiguous, and some senses are much more likely to be used reflexively than others. In some cases, some senses are inherently reflexive, while others are not. The senses of *opmaken* illustrated in (6a) and in (6b), for instance, can hardly be used reflexively, the sense in (6c) can easily be used with a reflexive, while the sense in (6d) is inherently reflexive.

(6) a. De bedrijven maakten foute rekeningen op
The companies produced wrong bills

b. De schelpdieren maken al het voedsel op
The shellfish take all the food

c. Als ik 240 rijd, kan mijn assistente zich rustig opmaken
If I drive 240, my assistent can put make-up on easily

[6]The examples in the section are not intended as a stand alone introduction into XPath and XQuery. Readers who want to know more are advised to consult Walmsley (2007) or on-line tutorials such as `http://en.wikibooks.org/wiki/XQuery`.

[7]http://saxon.sourceforge.net

```
<?xml version="1.0" encoding="UTF-8"?>
<results>
   <verb root="sluit_aan" sc="part_np_pc_pp(aan,bij)"/>sc
   <verb root="voel" sc="transitive_ndev_ndev"/>
   <verb root="sleep_mee" sc="ninv(np_ld_pp,part_np_ld_pp(mee))"/>
   <verb root="bijt" sc="transitive"/>
   <verb root="meld_af" sc="ninv(transitive,part_transitive(af))"/>
   <verb root="pers_samen" sc="part_transitive(samen)"/>
   <verb root="heb" sc="transitive_ndev"/>
   <verb root="stel" sc="np_ld_pp"/>
   <verb root="omarm" sc="transitive"/>
   <verb root="noem" sc="pred_np"/>
   <verb root="wurg" sc="transitive"/>
   <verb root="dring" sc="np_ld_pp"/>
   <verb root="dien_aan" sc="ninv(transitive,part_transitive(aan))"/>
   <verb root="sus" sc="transitive"/>
   <verb root="wapen" sc="transitive"/>
   <verb root="sta_bij" sc="ninv(transitive,part_transitive(bij))"/>
   <verb root="presenteer" sc="transitive"/>
   <verb root="moderniseer" sc="transitive"/>
</results>
```

Figure 6: Output of the XQuery script in figure 5.

d. De showbizz maakt zich op voor het huwelijk van het jaar
The showbizz prepares itself for the marriage of the year

Obviously, counting the frequency with which a verb occurs with an nonreflexive or reflexive object, without taking these differences in meaning into account, leads to noisy results. On the other hand, the parser does not annotate word senses, so we cannot automatically produce counts per verb sense.

As an approximation of word senses, we can use the `sc` feature. Often, different word senses correspond with differences in subcategorization frame as well. The inherent reflexive use of *opmaken* (6d), for instance, can be distinguished from the other senses by the fact that it subcategorizes for a PP-complement headed by the preposition *voor*. Another example is *omringen*, which has two (related) senses:

(7) a. Een leger adviseurs omringt professionele sporters
An army advisors surrounds professional athletes
b. De mountainbikewereld omringt zich met allerlei bedrijven
The mountainbike-world surrounds itself with all-kinds-of companies

The first sense subcategorizes for an NP only, whereas the second subcategorizes for a *met*-PP as well. The second sense is much more often used used with a reflexive than the first sense.

```
<results>
{ for $node in
     collection('ad19990104')/alpino_ds//node[@pos="verb" and ../node[@rel="obj1"]]
  let $obj := $node/../node[@rel="obj1"]
  let $obj-type :=
    if   ($obj/@root = "zich")
    then "zich"
    else  if   ($obj/@root = "zichzelf")
          then "zichzelf"
          else "np"
  return
     <verb obj-type="{$obj-type}">
       {$node/@root}
       {$node/@sc}
     </verb>
}
</results>
```

Figure 7: Extracting verbs selecting a direct object and returning the type of the selected object.

3.2 Selecting all NPs

The next step is to develop a script that matches with all occurrences of a verb with a direct object, and then returns the type of the direct object: *zich*, *zichzelf*, or any other NP. Our first attempt is in figure 7. We use `let` to define additional variables (mainly for readability). First, we introduce a variable for the direct object, and next we define the value of the obj-type variable by means of two if-then-else statements. Example output is given in figure 8

We now have the basics of an XQuery script that would count occurrences of a verb with a direct object, and which returns some information on the type of the object. The script still needs improvement, however, for a number of reasons:

1. There are a number of contexts in which a verb occurs with a direct object, but the

```
<?xml version="1.0" encoding="UTF-8"?>
<results>
 <verb obj-type="np" root="weer" sc="transitive"/>
   <verb obj-type="np" root="zie" sc="fixed([[schoon],acc(kans)],no_passive)"/>
   <verb obj-type="np" root="stel" sc="fixed([[ter,discussie],acc],norm_passive)"/>
   <verb obj-type="np" root="schrijf_op" sc="part_transitive(op)"/>
   <verb obj-type="zich" root="sluit_aan" sc="part_np_pc_pp(aan,bij)"/>
   <verb obj-type="zichzelf" root="zet_neer" sc="part_transitive(neer)"/>
   ....
</results>
```

Figure 8: Sample output of the XQuery script in figure 7.

direct object can never be a reflexive pronoun. The most obvious cases are passives (where the passive participle has a direct object dependent that is co-indexed with the subject). These cases should also be excluded from the counts.

2. For each verb, we want to count how often it occurs with *zich*, *zichzelf*, or any other NP. However, if we want to use the counts to estimate how likely it is that the verb occurs with a weak or strong reflexive pronoun, we should only count 'other' NPs in case the subject of the verb is third person, as only such NPs could ever occur with *zich* or *zichzelf*.

3. In the Alpino/LASSY XML, some direct objects are just index nodes. The content of such a node can only be found by locating the full node somewhere else in the dependency tree/graph.

4. The output is XML, but for postprocessing it can actually be more convenient to produce plain text as output.

5. The script is designed to work on a single directory. Our actual corpus is much larger, and processing such large data-sets requires a somewhat different set-up.

We discuss each of these issues below.

3.3 Selecting relevant cases

We decided to skip all occurrences of verbs that are used in passive sentences, or as complement of *laten*:

(8) a. De opstandelingen werden ontwapend
The rebels were disarmed
b. De kinderen laten zich niet dwingen
The children do not let themselves be forced
c. De kinderen laten zich vallen
The children let themselves fall

In passives, the object of the main verb appears as the subject of the passive auxiliary (see figure 9). In this position reflexives cannot be used. In sentences with *laten* (8b), a reflexive may appear as the object of the embedded verb (see figure 10). This reflexive is interpreted as coreferential with the subject of *laten*, but it is not coreferential with the subject of the embedded verb (*dwingen*). Therefore, it seems incorrect to count such examples as reflexive uses of the embedded verb. Note that there is also a use of *laten* + *zich* where *zich* is interpreted as a direct object of *laten* (8c). These are included, but in this case as examples of reflexive use of the verb *laten*.

In nominalizations, reflexives can only occur if the verb is inherently reflexive:

(9) a. Goede vragen verzinnen is moeilijk
good questions make-up is hard
b. Zich vergissen is menselijk
REFL mistake is human
c. *Zich(zelf) verbeteren is onmogelijk
REFL improve is impossible

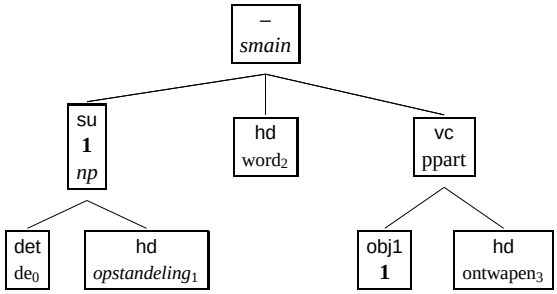

Figure 9: De opstandelingen werden ontwapend

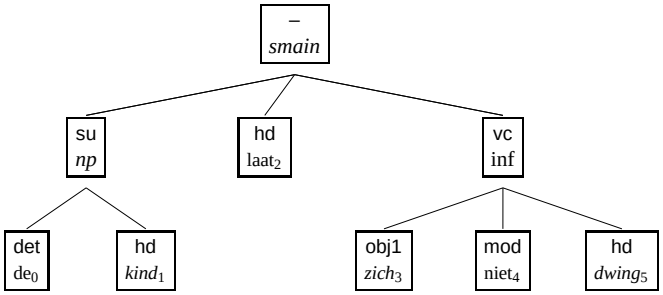

Figure 10: De kinderen laten zich niet dwingen

As a reflexive can normally not occur in a nominalization, these cases are discarded as well.

We introduce the script that skips passives, *laten*-constructions, and nominalizations in section 3.5.

3.4 Restricting the script to third person cases

As we only collect counts for the third person reflexives *zich* and *zichzelf*, which necessarily have to co-occur with a third person subject, it seems appropriate to restrict cases where a verb occurs with a non-reflexive object to cases where there is a third person subject as well.

For finite verbs, it seems this can simply be achieved by looking at the `infl` attribute. The `infl` attribute does not always distinguish between third person uses and other uses, however. Past tenses, for instance, are `past(sg)` or `past(pl)`, but do not make the person distinction. There are also verbs with irregular morphology, where the person distinction is absent in the present tense.

As an alternative, we can look at the `per` attribute of the subject. Not all NPs contain a head that has a per attribute. For first and second person pronouns, however, the `per` attribute always has a value `fir, sec, je` or `u`. By filtering these cases, we will ensure that only verbs with a third person subject are included in our counts.

3.5 Using the Alpino Module

Figure 12 gives the script which skips verbs in passive and laten constructions, nominalization contexts, and verbs with a non-third person subject. The first two cases can be identified

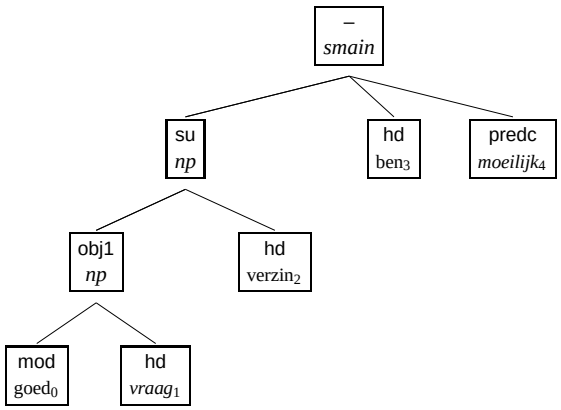

Figure 11: Goede vragen verzinnen is moeilijk

by looking at the *governing* verb, which in the XML encoding of a dependency tree is a `node[rel='hd']` that is a sister of the mother of the current node. Passive auxiliaries (*worden* or *zijn*) have an `sc='passive'` attribute, and *laten* can simply be identified by its root form.[8] Nominalizations are excluded by requiring that the verb has a subject dependent.

Non-third person subjects, finally, are excluded by looking at the `per` attribute of the subject. Note that we introduce the `alpino` XQuery module at this point. The function `head-of` is defined in the `alpino` module. It locates the grammatical head of a possibly complex constituent. Furthermore, in cases where a node is actually an index node, it first locates the fully specified node that is co-indexed with this index node. The relevant parts of the alpino module are given in figure 13.

A module can be loaded by including a module declaration as in the first line of figure 12. The location of the module in this case is a url, but it can also be a path to a local file. The module is loaded into the namespace `alpino`, and functions from the module therefore have to be prefixed with `alpino:`. More information on the alpino module can be found in Bouma and Kloosterman (2007).

3.6 Resolving Index nodes

If the object of a verb is an index node (i.e. a node co-indexed with a node somewhere else in the tree), we need to inspect the co-indexed, full, node, to determine the type of the NP. With objects, this happens mainly in questions where the object is a WH-NP, and in relative clauses where the object is the relative pronoun (see figure 14).

Dealing with such cases can be easily achieved with the `resolve-index` function of alpino. Instead of

```
let $obj := $node/../node[@rel="obj1"]
```

we now write

```
let $obj := alpino:resolve-index($node/../node[@rel="obj1"])
```

[8] Note that we cannot identify passive particple verbs by means of the `infl` attribute, as this does not distinguish between passive and perfect participles.

```
import module namespace
        alpino = "alpino.xq" at "http://www.let.rug.nl/gosse/alpino.xq" ;

<results>
{ for $node in
     collection('ad19990104')/alpino_ds//node[
                 @pos="verb"
              and ../node[@rel="obj1"]
              and ../node[@rel="su"]
              and not(../../node[@rel="hd" and (@sc="passive" or @root="laat")])
              and alpino:head-of(../node[@rel="su"])[not(   @per="fir" or @per="sec"
                                                         or @per="je"  or @per="u"  )]
              ]
  let $obj := $node/../node[@rel="obj1"]
  let $obj-type :=
    if   ($obj/@root = "zich")
    then "zich"
    else  if   ($obj/@root = "zichzelf")
          then "zichzelf"
          else "np"
  return
     <verb obj-type="{$obj-type}">
       {$node/@root}
       {$node/@sc}
     </verb>
}
</results>
```

Figure 12: Selecting only relevant cases.

```
module namespace alpino="alpino.xq" ;

(: resolve-index
   for index-only nodes return co-indexed non-empty node
   else return node itself
:)

declare function alpino:resolve-index($constituent as element(node)) as element(node)
  {   if   ( $constituent[@index and not(@pos or @cat)] )
      then  $constituent/ancestor::alpino_ds/descendant::node[@index = $constituent/@index
                                                              and (@pos or @cat)]
      else  $constituent
  };

(: head-of
   identify the lexical head node of a constituent:
   resolve index nodes, return the head (or similar) daughter,
   take 1st element of multiple crd daughters (of...of...)
   if no suitable hd daughter is found, return node itself
:)

declare function alpino:head-of($constituent as element(node)) as element(node)
  {  let $resolved := alpino:resolve-index($constituent)
     let $head := $resolved/node[   @rel="hd" or @rel="crd" or @rel="cmp"
                                 or @rel="dlink" or @rel="rhd" or @rel="whd" ]
     return
       if ($head) then
          $head[1]
       else $resolved
  };
```

Figure 13: Parts of the Alpino module located at `www.let.rug.nl/gosse/alpino.xq`

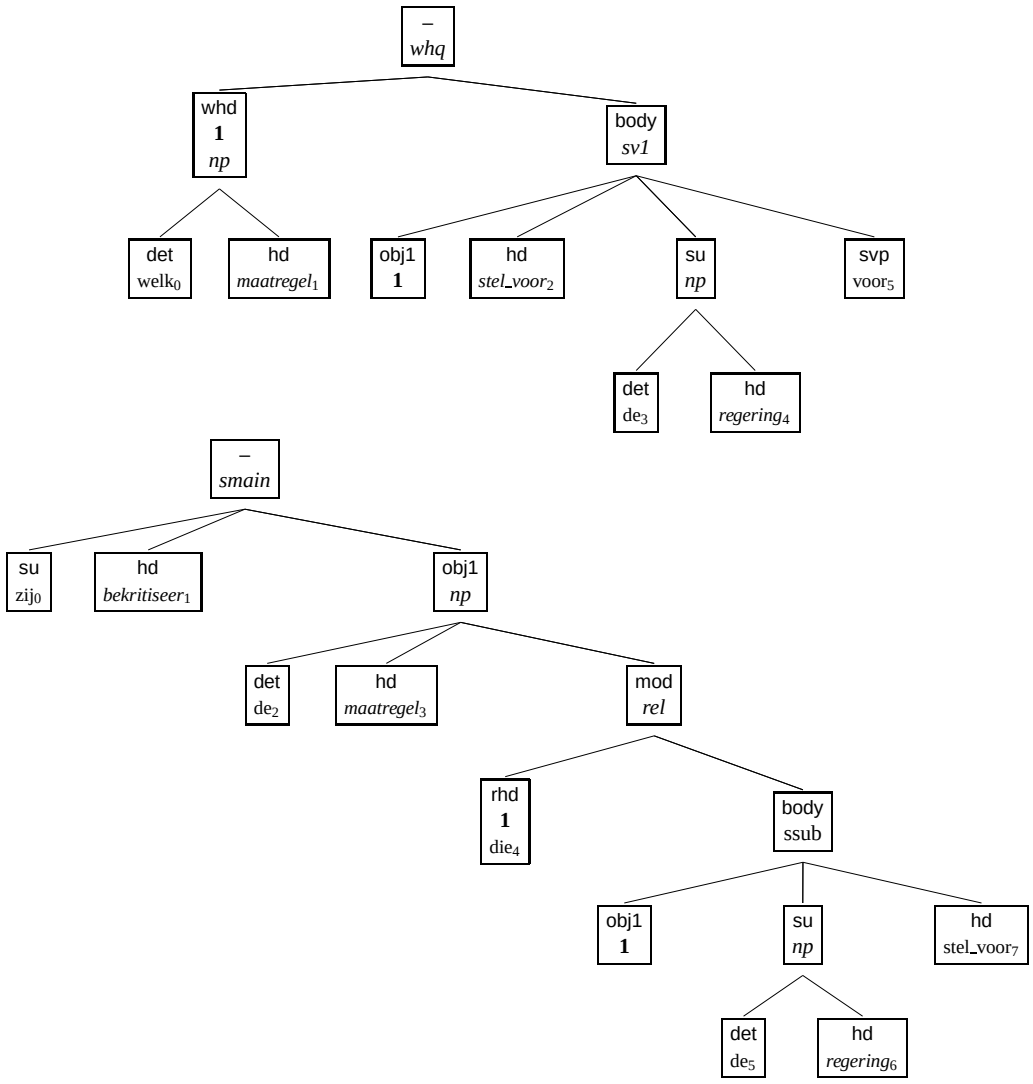

Figure 14: In WH-questions and relatives, the object can be an index node.

3.7 Producing text based output

The output of the script as it stands produces and XML document. This is a generic, yet somewhat verbose output format. For processing the data with scripting languages such as Perl, it is often convenient to provide output in a text based format, where the information about each verb is reprented in some fixed format on a single line.

XQuery can easily produce other outputs besides XML as well. We can change the return statement as follows:

```
return
 string-join(($obj-type,$node/@root,concat($node/@sc,'&#10;'),"#")
```

The `concat` function concatenates strings, and the `string-join` function concatenates strings but takes an additional second argument, that is used as separator. The XML entity `
` stands for a carriage return.

This produces the following kind of output:

```
np#kom_tegen#part_transitive(tegen)
np#onttrek#np_pc_pp(aan)
np#vraag#so_pp_np
np#koop#transitive
np#lijd#transitive
zich#moderniseer#transitive
```

As the script no longer produces XML, the outer `<result>` element is superfluous as well, and can be removed.

3.8 Scaling to large corpora

The results reported in Bouma and Spenader (2009) used data from the 470M word Twente News Corpus (TwNC), made up of the text of Dutch newspapers from the period 1994-2005 (Ordelman et al., 2007), which was parsed automatically with the Alpino-parser. An improved version of this corpus is released as the LASSY Large corpus. We searched the corpus exhaustively for all occurrences of a verb with an object and a third person subject, and registered whether the object was *zich, zichzelf*, a (non-reflexive) pronoun, or a regular NP. We extracted 12M verb-object tuples.

Working with corpora of this size is hardly feasible if the dependency trees for each sentence are stored individually on disk. To allow for efficient storage and access, the XML is compressed using dictzip. The tools are documented on the Alpino tools webpage.[9]

Tools for running an xquery script, such as saxon, normally expect XML data as input, stored as a file on disk or accessible by means of a url. To make the files accessible for saxon, we developed the following solution:

1. A file-server, which has the capability of providing the content of file ids specified by a client.

[9] `http://www.let.rug.nl/~vannoord/alp/Alpino/TreebankTools.html`

2. An xquery-client, which sends file ids to the server, and then sends the contents of the returned files to saxon for processing by means of a specific xquery script.

The advantage of the client-server architecture is twofold. First, files do not have to be (uncompressed and) stored on disk before they can be processed, and second, the client needs to start only a single saxon process, which can be used to process a stream of XML documents.[10]

The scripts presented earlier work in this set-up as well, except for the fact that in the `for` loop the collection is no longer explicitly mentioned. Instead of

```
for $node in collection('ad19990104')/alpino_ds//node[@pos="verb" and ...]
```

we now write

```
for $node in alpino_ds//node[@pos="verb" and ...]
```

A typical set-up is one where treebank-server is started as follows:

```
treebank-server -l -c /path/to/Treebank/directory
```

The `-l` flag means that `localhost` is used, the `-c` flag specifies the directory on the local machine where the treebank is stored. We can now start a different process that uses the xquery client:

```
dtlist -r /path/to/Treebank/directory  | xqclient -s zichzichzelf.xq
```

The `dtlist` command (part of the Alpino tools) lists the filenames of all files in a dictzip archive. These are simply piped to the xqclient. The `-s` flag specifies the script that is used to process the files. Normally, one would pipe the results to a file. It is our experience that the processing the full 470M word corpus and extracting 12M facts requires approximately 12 hours.

3.9 The actual script

For the experiments in Bouma and Spenader (2009) we extended the script with more functionality. The complete script can be found in Appendix A. One of the differences is that we collected more fine-grained data on NP-type, in case the NP was not a reflexive. In particular, we recorded the occurrence of (personal) pronouns. In the paper, it is argued that the ratio of (non-reflexive) pronoun use vs. reflexive use actually is a better predictor of strong reflexive use that non-reflexive argument vs reflexive use.

Furthermore, the script records occurrences of the reciprocal reflexive *elkaar*, the number of the subject, the presence of a focus particle such as *alleen* preceding zich or zichzelf, and the relative order of the subject and the reflexive pronoun. Although this information is of some interest for future research, it was not used in Bouma and Spenader (2009).

4 Postprocessing

From the output of the Xquery script, it is relatively easy to collect counts for individual words. For instance, for the verbs *straffen, beschermen* and *vastketenen*, we obtain the counts shown in table 2.

[10] *Automatic Extraction of Hypernymy Information* (Lassy Case Study I, WP 6.2), section 3.6, discusses a slightly different solution, using the treebank took `dz2saxon`. Note, however, that dz2saxon passes all files as a single XML document to saxon, and thus is restricted to document collections that can be handled in memory.

verb	nonrefl		refl		*zich*		*zichzelf*	
	#	%	#	%	#	%	#	%
straf (*to punish*)	1060	95.7	47	4.3	2	4.2	45	95.8
bescherm (*to protect*)	4921	96.4	186	7.6	95	51.1	91	48.9
vastketenen (*to chain*)	24	34.8	45	65.2	43	95.6	2	4.4

Table 2: Counts and percentages for nonreflexive and reflexive use, and use of weak and strong reflexive pronouns.

```
PercStr RatioStr LogRatStr CountStr PercRef RatioNRefl LogRatNRefl CountNRefl Key
33.33   0.50     -0.6931   (6/12)   76.00   3.17       1.1527      (57/18)    klop#np_ld_pp#obj1
1.43    0.01     -4.2341   (1/69)   40.17   0.67       -0.3983     (47/70)    sleep#transitive#obj1
81.12   4.30     1.4579    (1620/377) 37.77 0.61       -0.4994     (1212/1997) zie#als_pred_np#obj1
2.63    0.03     -3.6109   (3/111)  1.72    0.02       -4.0431     (2/114)    matig#transitive#obj1
```

Table 3: Statistics for the XQuery output, produced using a Perl script, and suitable for processing with R.

For systematic analysis of the results, we need to have tables that list such percentages for all verbs in the corpus. We wrote a Perl script to do the necessary counting and computations. The script is included in Appendix B. For each verb + subcategorization value, it returns output as shown in table 3. It not only computes percentages, but also ratios, and log ratios.

The output of the Perl script was made so that it can be used as input for the statistical analysis package R.

The first line of figure 15 shows how a data file zich.ratio can be loaded in R.[11] Its contents is stored as the data frame `refl`. This object can be inspected by means of the `head` command, which displays the first rows of the data frame. The `density` function computes the density curve for the distribution of the values in a given column of the data frame. The `plot` function displays the density curve in a separate window, as shown in figure 16. Plots like this can also be saved in a file, The `pdf` function creates a pdf file. The next plot command now does not open a new window, but its output is redirected to the file. The `dev.off` command closes the file. The result of plotting the log values of the ratio of nonreflexive over reflexive use is shown in figure 17. Many more visualisation options are described in Baayen (2008).

We used linear regression to determine to what extent there is a correlation between reflexive use of a (non-inherent reflexive) verb and the relative preference for a weak or strong reflexive pronoun. The `lm` function (for *linear modelling*) in figure 18 estimates a linear function for the correlation between the log ratio of reflexive over nonreflexive use, and the log ratio of strong reflexive over weak reflexive use. The main properties of the result are given by the `summary` function.[12] Next, we make a plot of the individual data points, and the function computed by lm. The result in in figure 19.

[11] `http://cran.at.r-project.org`

[12] The R^2 value of 0.2927 and the standard error of 1.977 indicates that there is a weak correlation between the two values.

```
> refl = read.table('zich.ratio', header=TRUE)
> head(refl)
  PercStr RatioStr LogRatioStr   CountStr PercNonRefl RatioNonRefl
1   33.33     0.50     -0.6931     (6/12)       93.55        14.50
2    8.16     0.09     -2.4204     (4/45)        3.92         0.04
3    1.67     0.02     -4.0751    (7/412)        3.90         0.04
4    1.43     0.01     -4.2341     (1/69)       60.23         1.51
5   81.12     4.30      1.4579 (1620/377)       62.41         1.66
6   60.00     1.50      0.4055      (9/6)       75.81         3.13
  LogRatioNonRefl CountNonRefl         Key
1          2.6741     (261/18)        klop
2         -3.1987       (2/49) moderniseer
3         -3.2047     (17/419)   stel_open
4          0.4149     (106/70)       sleep
5          0.5071  (3316/1997)         zie
6          1.1421      (47/15)     lanceer
> plot(density(refl$LogRatioNonRefl))
> pdf("distribution.pdf")
> plot(sort(refl$LogRatioNonRefl))
> dev.off()
```

Figure 15: Loading data into R and visualising distributions

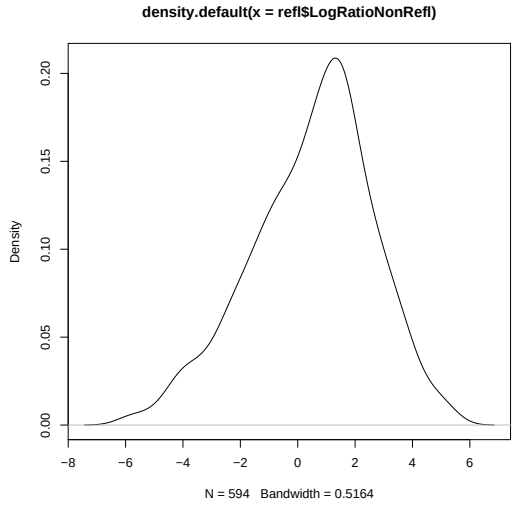

Figure 16: Density curve for the log values of the ratio of reflexive over nonreflexive use

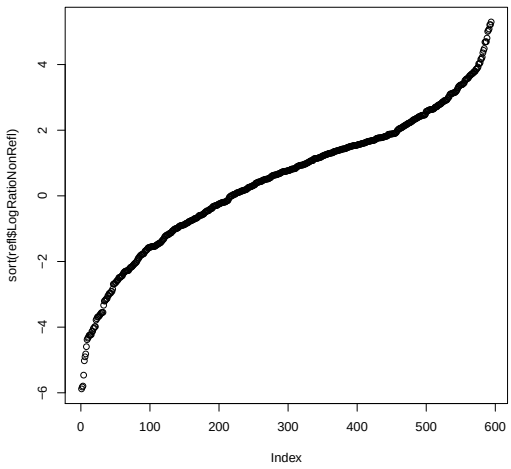

Figure 17: Distribution of the log values of the ratio of reflexive over nonreflexive use

5 Discussion of Linguistic Results

5.1 Distribution of Zich and Zichzelf

For accidental reflexive verbs in general, the use of *zich* was more frequent than *zichzelf*. We find 163K (84%) occurrences of *zich* vs. 31K (16%) occurrences of *zichzelf*. For more detailed observations, we restrict attention to verb+subcategorization pairs that occur at least 50 times in the corpus, and at least 10 times with a reflexive (899 cases, of which, according to the grammar, 163 are inherent reflexive verbs, and 736 are accidental reflexive verbs). If we restrict attention to reflexive use of transitive verbs, on average the reflexive *zich* is used 64.2% of the time, and the strong reflexive *zichzelf* is used 35.8% of the time. Although *zichzelf* in general is rare, we find that 6% of the accidental reflexive verbs (44 of 736) , when used reflexively, occur with a strong reflexive more than 95% of the time. Examples are *zichzelf in de weg zitten (hinder oneself), toespreken (address), opvoeren als (present), tegenkomen (encounter), onderschatten (underestimate), kwijtraken (loose), bedriegen (cheat), haten (hate), ombrengen (kill), ervaren als (experience), gebruiken als (use as), uitnodigen voor (invite for), afschrijven (write off),* and *onderbreken (interrupt).* 34% of the accidental reflexive verbs (247) occur with a strong reflexive more than 50% of the time, which is considerably over the average of 35.8%. 25% of the accidental reflexive verbs (187) occur with a strong reflexive less than 8% of the time (less than a quarter of the average of 35.8%). Some examples of the latter group are *beheersen (withhold), voorstellen (introduce), manoeuvreren (manoevre), uitleveren (hand over to), bevrijden (liberate), wassen (wash), aankleden (dress), scheren (shave), beschikbaar stellen (make available).* We do find a number of 'outward directed' verbs among the group of verbs with a strong preference for *zichzelf*, and a number of 'self directed' verbs in the group with a dispreference for *zichzelf.* This is in line with Haspelmath's semantic characterization of such verbs.

The 44 verbs with a strong preference for the strong reflexive *zichzelf* were used non-reflexively 97.1% of the time. The 247 verbs used more often with a strong reflexive than

```
> refl.lm = lm(LogRatioStr ~ LogRatioNonRefl, data = refl)
>  summary(refl.lm)

Call:
lm(formula = LogRatioStr ~ LogRatioNonRefl, data = refl)

Residuals:
    Min      1Q  Median      3Q     Max
-5.0762 -1.4058 -0.1613  1.3925  5.6490

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)     -1.48388    0.08323  -17.83   <2e-16 ***
LogRatioNonRefl  0.59948    0.03830   15.65   <2e-16 ***
---
Signif. codes:  0 ?***? 0.001 ?**? 0.01 ?*? 0.05 ?.? 0.1 ? ? 1

Residual standard error: 1.977 on 592 degrees of freedom
Multiple R-squared: 0.2927,     Adjusted R-squared: 0.2915
F-statistic: 244.9 on 1 and 592 DF,  p-value: < 2.2e-16
> plot(refl$LogRatioStr ~ refl$LogRatioNonRefl)
> abline(refl.lm)
```

Figure 18: Computing and visualizing correlation

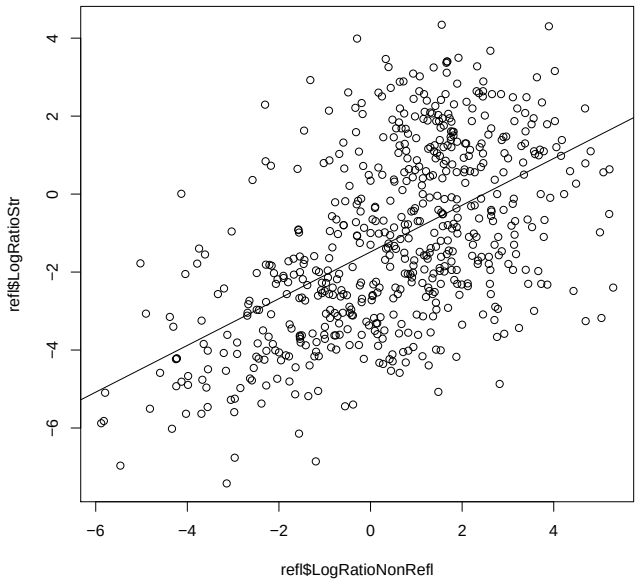

Figure 19: Correlation between nonreflexive use and strong reflexive use

with a weak reflexive were used non-reflexively 95.1% of the time. The 187 verbs used with a strong reflexive less than 8% of the time were used non-reflexively 72.0% of the time. This suggests that there is indeed a relationship between preference for the strong reflexive form and a high relative frequency of non-reflexive use.

Traditionally, it is claimed that inherent reflexives never occur with the strong reflexive *zichzelf*. We can examine empirically whether or not this is in fact true. Of the 163 reflexive verbs in our data-set, 112 (68.7%) occur with *zich* more than 99% of the time (often with only 1 or 2 occurrences of *zichzelf*).[13]. The frequency for such usage is too low to be reliable, and the examples could very well include parsing errors. However, 51 inherent reflexives occurred relatively frequently with strong reflexive objects. Here are a number of examples:

(10)
a. Nederland moet stoppen zichzelf op de borst te slaan
The Netherlands must stop beating itself on the chest
b. Hunze wil zichzelf niet al te zeer op de borst kloppen
Hunze doesn't want to knock itself on the chest too much
c. Ze verloren zichzelf soms in tactische varianten
They lost themselves in tactical variants
d. Ze verliezen zichzelf niet in slaapverwekkende dialogen
They don't lose themselves in boring dialogues
e. Hij verbeeldt zichzelf oogcontact te hebben
He imagines himself to have eye contact
f. Met de hulp van dieren weet hij zichzelf te vermannen
With the help of animals, he gives himself new courage
g. Serieus nu, vermant Wyclef Jean zichzelf
But seriously, Wyclef Jean gives himself new courage
h. Laat ik er maar trots op zijn, nam hij zichzelf voor
I should better be proud, he promises himself
i. Reeds voor het EK nam hij zichzelf voor op te stappen
Already before the European Championships, he promised himself to quit
j. Eerst achter de jongens aan, nam Bellaart zich THIS IS A ZICH! voor
First follow the boys, Bellaart promised himself
k. Bomans prees zichzelf gelukkig dat hij bij haar thuis mocht komen
Bomans praised himself lucky that he could visit her at all
l. Bush zei dat McVeigh zichzelf gelukkig mag prijzen dat hij in de VS leeft
Bush said McVeigh should praise himself lucky to live in the US

The idiomatic expression *zich/zichzelf op de borst kloppen* (*to boast*) occurs with a strong reflexive 47 times (30% of the time). A few other idiomatic expressions behave similarly. One explanation might be that the idiomatic readings are still transparently linked to the non-idiomatic, accidental reflexive, reading, leading to a certain amount of interference between the two uses.

[13]Note that ambiguous cases such as *bedruipen*, which has both an inherently reflexive form and an accidental reflexive form, were excluded from our counts.

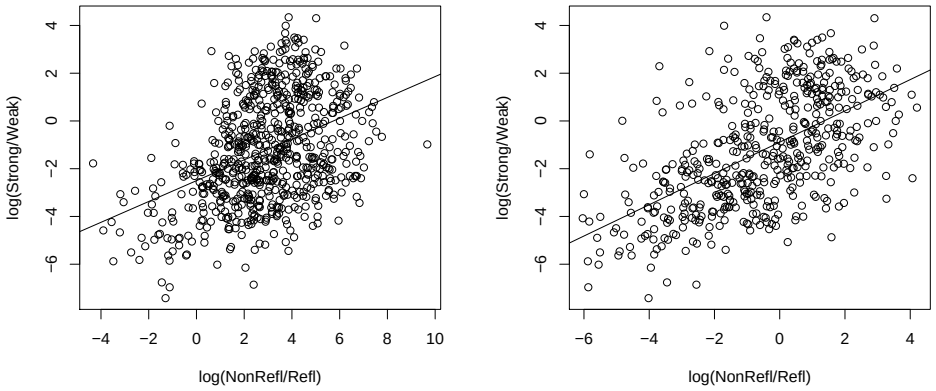

Figure 20: Nonreflexive vs reflexive use compared with strong reflexive over weak reflexive use counting all NP-objects (left) and counting only pronouns (right).

5.2 Statistical Analysis

As before, we limit our 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. Figure 20 (left pane) plots the ratio of nonreflexive use over reflexive use (x-axis) against the ratio of strong reflexive forms over weak reflexive forms (y-axis) for all objects. Linear regression (shown as the solid line in fig. 20) gives an r^2 correlation coefficient of 0.162 (statistically significant at $p < 0.001$), with a standard error of 2.07. This means that the ratio of nonreflexive over reflexive use accounts for 16% of the variance in the ratio of strong reflexive over weak reflexive use.

If we count as non-reflexive uses only cases where a verb occurs with a pronoun (as suggested by Haspelmath), 594 verbs remain with frequencies above the cut-offs we used. Linear regression over this data set gives an r^2 of 0.293, and a slightly lower standard error (1.98). If we only consider third person personal pronouns only (*hem (him), haar (her), hen (them)* and *ze (them)*), 500 verbs remain. We now obtain the result given in fig. 20 (right pane), with an r^2 of 0.332 and a standard error of 1.97.

These results are in line with the findings in Hendriks, Spenader, and Smits (2008). They also observed that restricting object counts to personal pronouns gives a better result than counting all NP-objects. However, for the 32 verbs for which they collected data, they obtain an r^2 of 0.456. As we obtain an r^2 of 0.332, the question arises what might explain this difference. We extracted all verbs from the data-set for personal pronouns that were also used in Hendriks, Spenader, and Smits (2008). 24 of these verbs were sufficiently frequent in our data-set. Linear regression over this limited set gives an r^2 of 0.547 and a standard error of 1.7. One reason for the higher score (compared to Hendriks *et al.*) might be the fact that we take subcategorization frames into account. Another reason might be our use of different frequency cut-offs. What the result also shows, is that our method of data collection in itself does not introduce more noise than the method in Hendriks, Spenader, and Smits (2008). The fact that we obtain a lower score on the larger set of verbs could be due to the fact that the 32 verbs used by Hendriks, Spenader, and Smits (2008) were collected from examples used in the

literature. Apparently, these verbs are particularly suitable for demonstrating the statistical correlation to be investigated. Once one takes the full set of verbs into account, however, a fair number of outliers is added as well.

5.3 Discussion

One of the major ways in which this work tries to improve upon earlier work is by using more data, looking at more verbs (hundreds rather than 30-50) and by using better data (by distinguishing verbs by their subcategorization frames). The assumption is that more data will lead to a better model, and will compensate for irregularities introduced by the fully automated process. Looking at more data did lead to higher correlations for each of the data collection methods, though this effect is not distinguishable from the effect of separating verbs by subcategorization frame.

But looking at more verbs did not give higher correlations. The highest correlation was obtained with the verbs studied by Hendriks, Spenader, and Smits (2008). These are verbs that routinely appear in the literature as good examples of accidental reflexives. One explanation is that these verbs are relatively frequent (although not necessarily frequent in our corpus), and that frequent verbs are the ones for which a speaker may have an expectation of self-directedness or other-directedness. Another explanation is that these verbs in particular might have relatively few different senses, or that they are overwhelmingly used with a sense that has the potential to be both self- or other-directed.

It is still not clear why the ratio of pronominal objects to reflexive objects predicts so much better than taking all objects into account. There are two possible explanations. First, it may be that this restriction in a way also filters out uses of verbs with senses that essentially cannot be used reflexively. By only counting pronominal objects as non-reflexive objects, the sense of the verb has to be one where the action can be performed on another agent. This would lead to more accurate data (though less data) and may be responsible for the better results.

The other explanation comes from theoretical syntax, Principle A and B of the Binding Theory (Chomsky, 1981) suggests that personal pronouns and reflexives are in complementary distribution when the subject and the object are both animate. In other words, there is a potential for reflexive action only in the case of an animate subject. This means that the ratio for a given action to be self- or other-directed is only reliable if we limited our counts to cases where the subject and object are both animate.

Strictly speaking, comparing the ratio of pronominal objects to reflexive objects does not actually give us the ratio of self- vs. other-directed events. This is because we also potentially count cases where the subject is inanimate and the object is a personal pronoun. However, the few corpus studies of grammatical role and animacy that have been done show that the combination of an inanimate subjects with an animate objects is disprefered. Bouma (2008) gives results for spoken Dutch with data for 2345 sentences from the *Corpus Gesproken Nederlands*. 243 of the sentences had animate objects but among these only 8 (or 3%) occurred with an inanimate subject. Using data from written texts, Ovrelid (2004) looked at 1,000 randomly sampled sentences from the Oslo corpus of Norwegian. 98 of the 1,000 sentences studied had animate objects and of these only 24 had an inanimate subject (24%).

Still, we are able to account for between 30-53% of the data (depending on what dataset is used) using only one predictive factor: how frequently the verb is used with a reflexive object. However, it is also clear that other factors play a role in choosing between a strong and

	zichzelf	*zich*		*zichzelf*	*zich*
alleen (only)	109	1	*nu (now)*	16	1
ook (also)	214	9	*wel (certainly)*	14	0
niet (not)	30	9	*min of meer (more or less)*	21	0
slechts (only)	2	*alleen maar (only)*	13	1	
zelfs (even)	7	0	*zo (that way)*	12	0

Table 4: Choice of reflexive immediately following focus particles

reflexive form. Only strong reflexives can be coordinated, fronted and phonetically focused. This suggests we should take such additional factors into account as well. But coordination of reflexives is rare, and focus or phonetic stress is hard to determine automatically. In a limited number of cases, one might try to determine focus by taking the preceding expression into account. If the word preceding the reflexive object is a focusing particle, we expect the reflexive following to be *zichzelf*. Table 4 shows that this is indeed the case for a number of expressions that associate with focus.

Factors such as position in the sentence could also be checked. For example, we expect only strong reflexives to be fronted, so we would expect more strong reflexives in initial sentence position. Further, because only strong reflexives can receive sentential accent we would also expect strong reflexives to occur sentence finally more often than weak reflexives (with accidental reflexive verbs). It would be interesting to collect data for the (relative) sentence position of the reflexive (i.e. distance (in words or constituents) from the governing verb or end of the sentence), and to investigate whether a correlation can be found between position and reflexive choice. Geurts (2004) suggests yet another factor. Even non-reflexive verbs like *toedienen* (*to inject oneself*) can use *zich* if the context makes clear the action is a habitual event. This suggests that the presence of temporal adverbs indicating frequency could also play a role. If we can find methods to collect the relevant data automatically, it would be interesting to incorporate them in a multivariate analysis in future work.

6 Conclusions

In this case study, we have shown how a linguistic problem can be studied using data from the LASSY Large corpus. We have introduced the XML format and some of the basic tools for working with the corpus. Next, we have discussed in some detail the use of XQuery for extracting more detailed data from the corpus, and for processing large corpora. We have briefly touched upon the subject of analyzing the results of data collection using R, and presented the main results of Bouma and Spenader (2009).

References

Baayen, R.H. 2008. *Analyzing Linguistic Data.* Cambridge University Press.

Bouma, Gerlof. 2008. *Starting a sentence in Dutch.* Ph.D. thesis, University of Groningen.

Bouma, Gosse and Geert Kloosterman. 2007. Mining syntactically annotated corpora using xquery. In Branimir Boguraev and Nancy Ide et al., editors, *Proceedings of the Linguistic Annotation Workshop (ACL 07)*, Prague.

Bouma, Gosse and Jennifer Spenader. 2009. The distribution of weak and strong object reflexives in dutch. In Frank van Eynde, Anette Frank, Koenraad de Smedt, and Gertjan van Noord, editors, *Proceedings of the seventh workshop on Treebanks and Linguistic Theory (TLT 7)*, Groningen.

Chomsky, Noam. 1981. *Lectures on Government and Binging.* Foris Dordrecht.

Geurts, Bart. 2004. Weak and strong reflexives in dutch. In *Proceedings of the ESSLLI workshop on semantic approaches to binding theory*, Nancy, France.

Haspelmath, Martin. 2004. A frequentist explanation of some universals of reflexive marking. Draft of a paper presented at the Workshop on Reciprocals and Reflexives, Berlin.

Hendriks, Petra, Jennifer Spenader, and Erik-Jan Smits. 2008. Frequency-based constraints on reflexive forms in dutch. In *Proceedings of the 5th International Workshop on Constraints and Language Processing*, pages 33–47, Roskilde, Denmark.

Ordelman, Roeland, Franciska de Jong, Arjan van Hessen, and Hendri Hondorp. 2007. Twnc: a multifaceted Dutch news corpus. *ELRA Newsletter*, 12(3/4):4–7.

Ovrelid, Lilja. 2004. Disambiguation of syntactic functions in norwegian: modeling variation in word order interpretations conditioned by animacy and definiteness. In *Proceedings of the 20th Scandinavian Conference of Linguistics*, Helsinki.

Smits, Erik-Jan, Petra Hendriks, and Jennifer Spenader. 2007. Using very large parsed corpora and judgement data to classify verb reflexivity. In Antonio Branco, editor, *Anaphora: Analysis, Algorithms and Applications*, pages 77–93, Berlin. Springer.

Walmsley, Priscilla. 2007. *XQuery.* O'Reilly.

A Extended XQuery script

```
import module namespace alpino = "alpino.xq"
                 at "/storage/gosse/Alpino/Treebank/Xquery/alpino.xq" ;

declare namespace saxon="http://saxon.sf.net/";
declare option saxon:output "omit-xml-declaration=yes";
declare option saxon:output "encoding=iso-8859-1" ;

(: determine the number of the subject by looking at the infl attribute of
   the closest governing finite verb (if any)
   this is preferred over using the num attribute of the subject NP itself,
   which often is 'both'
   looking at the finite verb fails in object-control cases:
        Ik blijf mensen oproepen zich tegen dierenmishandeling te verzetten .
:)

declare function local:su-number($verb as element(node)) as xs:string
  {if ( $verb[@pos="verb" and not(@infl="psp" or @infl="inf" or @infl="inf(no_e)" )] )
   then string($verb/@infl)
   (: finite verbal projections :)
   else if ( $verb/node[@rel="hd" and @pos="verb"
                        and not(@infl="psp" or @infl="inf" or @infl="inf(no_e)" )] )
   then string($verb/node[@rel="hd"]/@infl)
   (: report failure :)
   else if ($verb[@cat="top"])
   then "no-governing-finite-verb-found"
   (: report failure for (inf and te-inf) obj-control cases :)
   else if ($verb[@cat="inf" and node[@rel="su"]/@index = ../node[@rel="obj1"]/@index ] or
            $verb[@cat="inf" and node[@rel="su"]/@index = ../../node[@rel="obj1"]/@index ]
           )
   then "object-control-verb-found"
   (: else go up :)
   else local:su-number($verb/..)
  };

(: ignore following cases
   -- verb is not the head
      winteren deed het pas aan het einde van de maand
   -- both se and obj1 present (where se is like an obj2?)
      ik trok me zijn lot aan
   -- complement of laten (i.e. 'long-distance passive')
      de vissen laten zich aaien
   -- passives (i.e. subject can never be a reflexive)
      de vissen worden geaaid
   -- non-third-person subject cases
      * not all NPs have per attribute, so check for absence of per=fir/sec
:)

(: does u count as third person?? u hebt/heeft zich vernederd :)
for $node in /alpino_ds//node[@rel="hd" and @pos="verb"
             and count(../node[@rel="obj1" or @rel="se"]) = 1
             and not(../../node[@rel="hd" and (@sc="passive" or @root="laat")])
```

```
            and ../node[@rel="su"]
            and alpino:head-of(../node[@rel="su"])
                                [not(@per="fir" or @per="sec" or @per="je" or @per="u")]
            ]

let $root := alpino:head-root-string($node)

let $subcat :=
    (: normalize ninv(transitive,part_transitive(uit)) --> part_transitive(uit)
    :)
    replace(replace(replace(string($node/@sc),"_ndev",""),"\)\)",")"),"ninv.*part_","part_")


let $obj-node := $node/../node[@rel="obj1" or @rel="se"]

let $obj-rel := string($obj-node/@rel)

let $resolved-obj-node := alpino:resolve-index($obj-node)

let $obj-type:=
  if ($resolved-obj-node[@root="zich"] )           then "zich"
  else if  ($resolved-obj-node[@root="zichzelf"] ) then "zichzelf"
  else if  ($resolved-obj-node[@root="elkaar"] )   then "elkaar"
  else if  ($resolved-obj-node[@pos="pron"] )      then string($resolved-obj-node/@root)
  else if  ( not(alpino:neclass($resolved-obj-node)="nil") )
       then   alpino:neclass($resolved-obj-node)
  else if ($resolved-obj-node/@pos)                then string($resolved-obj-node/@pos)
  else string($resolved-obj-node/@cat)

let $alleen :=
  if ($resolved-obj-node[(@root="zich" or @root="zichzelf" or @root="elkaar")
                         and @begin=../node[@pos or @cat]/@end ])
  then alpino:yield($resolved-obj-node/../node[(@pos or @cat)
                                               and @end = $resolved-obj-node/@begin])
  else "nil"

(: count as su-zich only cases where su is not sentence initial
   ie follows vfin or is in subordinate clause
:)
let $zich-before-su :=
  if (     $node/../node[@rel="se" or @rel="obj1"
                         and (@root="zich" or @root="zichzelf" or @root="elkaar")]
      and $node/../node[@rel="su" and (@pos or @cat)]
     )
  then if (number($node/../node[@rel="se" or @rel="obj1"]/@begin) lt
           number($node/../node[@rel="su"]/@begin))
       then "zich-su"
       else if ( number($node/../node[@rel="su"]/@begin) gt 0 )
            then "su-zich"
            else "other"
  else "nil"
```

```
let $resolved-su-node := alpino:resolve-index($node/../node[@rel="su"])

let $su-type:=
  if  ($resolved-su-node[@pos="pron"] )      then string($resolved-su-node/@root)
  else if  ( not(alpino:neclass($resolved-su-node)="nil") )
       then   alpino:neclass($resolved-su-node)
  else if ($resolved-su-node/@pos)                then string($resolved-su-node/@pos)
  else string($resolved-su-node/@cat)




(: expand ad19990120-165-2-3.xml to twncn/COMPACT/AD1999/ad19990120/ad19990120-165-2-3.xml  :)

let $docid := replace(replace(base-uri($node),".*Machine/",""),".xml","")

return

concat($docid,"#",$root,"#",$subcat,"#",$obj-rel,"#",$obj-type,"#",$alleen,'#',
       $zich-before-su,'#',$su-type,'#',local:su-number($node),"&#10;")
```

B Perl script for processing XQuery output

```
#!/usr/bin/perl -w

use Getopt::Std;

#minimum number of events for a given verb
my $minfreq = 50 ;
#minimum number of reflexive uses for a given verb
my $minreflfreq = 10 ;
# count all objects or only pronouns
my $countpronouns = 0 ;
my $perc = 1 ;

getopts('f:r:phe') ;

$minfreq = $opt_f if ($opt_f) ;
$minreflfreq = $opt_r if ($opt_r) ;
$countpronouns   = 1 if ($opt_p or $opt_p) ;  # avoid single use error msg
$countpronouns   = 2 if ($opt_e or $opt_e) ;  # avoid single use error msg

if ($opt_h or $opt_h) {
die("Usage:
refl_nonrefl_comparison [OPTIONS] subcat.sorted
Options:
-f N minimum number of events for a given verb
-r N minimum number of reflexive uses for a given verb
-p   count only personal pronominal objects
-e   count all pronominal objects (extended)
-h   this message
") ;
}

if ($countpronouns eq 1) {
    print "Setting: at least $minfreq occurrences, $minreflfreq reflexives, " ;
    print "counting 3rd person personal pronouns\n"
;
} elsif ($countpronouns eq 2) {
    print "Setting: at least $minfreq occurrences, $minreflfreq reflexives, " ;
    print "counting all personal pronouns and person
names\n" ;
}
else {
    print "Setting: at least $minfreq occurrences, $minreflfreq reflexives, ";
    print "counting all objects\n" ;
}

my %reflexive = my %strongreflexive = my %reflall = () ;
my %all = () ;

my %personalpronouns = ( hem => 1, haar => 1, hen => 1, ze => 1 );
```

```
my %extendedpronouns =
  ( hem => 1, haar => 1, hen => 1, ze => 1,
    ieder => 1, me => 1, mij => 1, ons => 1,
    je => 1, u => 1, iedereen => 1, niemand => 1,
    wie => 1, jou => 1, hun => 1, mezelf => 1,
    jullie => 1, jezelf => 1, uzelf => 1, mijzelf => 1,
    haarzelf => 1, onszelf => 1, henzelf => 1,
    eenieder => 1, diegene => 1,
    PER => 1
  ) ;

while ($line = <>) {
    chop($line) ;
    $line =~ s/^ *// ;
    $line =~ /^([0-9]*) (.*)$/ ;
    my $count = $1 ;
    my $string = $2 ;
    my ($root,$subcat,$rel,$word) = split(/\#/,$string) ;
    my $key = join("\#",$root,$subcat,$rel) ;
    if (!($countpronouns) and $word !~ /^zich/ ) {
        $all{$key} += $count ;
    } elsif ($countpronouns eq 1 and exists $personalpronouns{$word} ) {
        $all{$key} += $count ;
    } elsif ($countpronouns eq 2 and exists $extendedpronouns{$word} ) {
        $all{$key} += $count ;
    }
    if ( $word =~ /zichzelf/ ) {
        $strongreflexive{$key} += $count ;
        $reflall{$key} += $count ;
        $all{$key} += $count ;
    } elsif ( $word =~ /zich/ ) {
        $reflall{$key} += $count ;
        $all{$key} += $count ;
    }
}

print "PercStr RatioStr LogRatioStr CountStr PercNonRefl RatioNonRefl " ;
print "LogRatioNonRefl CountNonRefl Key\n" ;

foreach  my $i (keys %strongreflexive) {  # require at least 1 occurrence of zichzelf
    my $strongrefl = my $refl = 0 ;
    if ( $all{$i} > $minfreq  and $reflall{$i} > $minreflfreq ) {
        if (exists $strongreflexive{$i} ) {
            $strongrefl = $strongreflexive{$i} ;
        }
        my $weakrefl = $reflall{$i} - $strongrefl ;
        my $nonrefl  = $all{$i} - $reflall{$i} ;
my ($root,$subcat,$rel) = split(/\#/,$i) ;
my $key2 = "xxx" ;
if ($subcat =~ /^transitive$/ ) {
   $key2 = join("\#",$root,"refl","se") ;
} elsif ($subcat =~ /^refl$/ ) {
   $key2 = join("\#",$root,"transitive","obj1") ;
```

```
} elsif ($subcat =~ /^part_transitive\((.*)\)$/ ) {
   $key2 = join("\#",$root,"part_refl($1)","se") ;
} elsif ($subcat =~ /^part_refl\((.*)\)$/ ) {
   $key2 = join("\#",$root,"part_transitive($1)","obj1") ;
} elsif ($subcat =~ /^np_pc_pp\((.*)\)$/ ) {
   $key2 = join("\#",$root,"refl_pc_pp($1)","se") ;
} elsif ($subcat =~ /^refl_pc_pp\((.*)\)$/ ) {
   $key2 = join("\#",$root,"np_pc_pp($1)","obj1") ;
} elsif ($subcat =~ /^part_np_pc_pp\((.*)\)$/ ) {
   $key2 = join("\#",$root,"part_refl_pc_pp($1)","se") ;
} elsif ($subcat =~ /^part_refl_pc_pp\((.*)\)$/ ) {
   $key2 = join("\#",$root,"part_np_pc_pp($1)","obj1") ;
} elsif ($subcat =~ /^np_ld_pp$/ ) {
   $key2 = join("\#",$root,"refl_ld_pp","se") ;
} elsif ($subcat =~ /^refl_ld_pp$/ ) {
   $key2 = join("\#",$root,"np_ld_pp","obj1") ;
} elsif ($subcat =~ /^part_np_ld_pp\((.*)\)$/ ) {
   $key2 = join("\#",$root,"part_refl_ld_pp($1)","se") ;
} elsif ($subcat =~ /^part_refl_ld_pp\((.*)\)$/ ) {
   $key2 = join("\#",$root,"part_np_ld_pp($1)","obj1") ;
} elsif ($subcat =~ /^ap_pred_np$/ ) {
   $key2 = join("\#",$root,"ap_pred_refl","se") ;
} elsif ($subcat =~ /^ap_pred_refl$/ ) {
   $key2 = join("\#",$root,"ap_pred_np","obj1") ;
}
if (exists $all{$key2} ) {
   print "AMBIGUOUS $i\n" ;
        } elsif ($weakrefl > 0 and $nonrefl > 0 ) {
            my $strongreflperc = 100 * $strongrefl/$reflall{$i} ;
            my $nonreflperc    = 100 * (1 - ($reflall{$i}/$all{$i})) ;
    my $strongreflratio = $strongrefl / $weakrefl ;
    my $nonreflratio    = $nonrefl/ $reflall{$i} ;
            my $key = $i ;
            $key =~ s/ /+/g ;
printf "%2.2f %2.2f %2.4f ($strongrefl/$weakrefl)\t%2.2f %2.2f %2.4f ($nonrefl/$reflall{$i})\t$key\n",
                  $strongreflperc, $strongreflratio, log($strongreflratio),
                  $nonreflperc, $nonreflratio , log($nonreflratio) ;
        }
    }
}
```